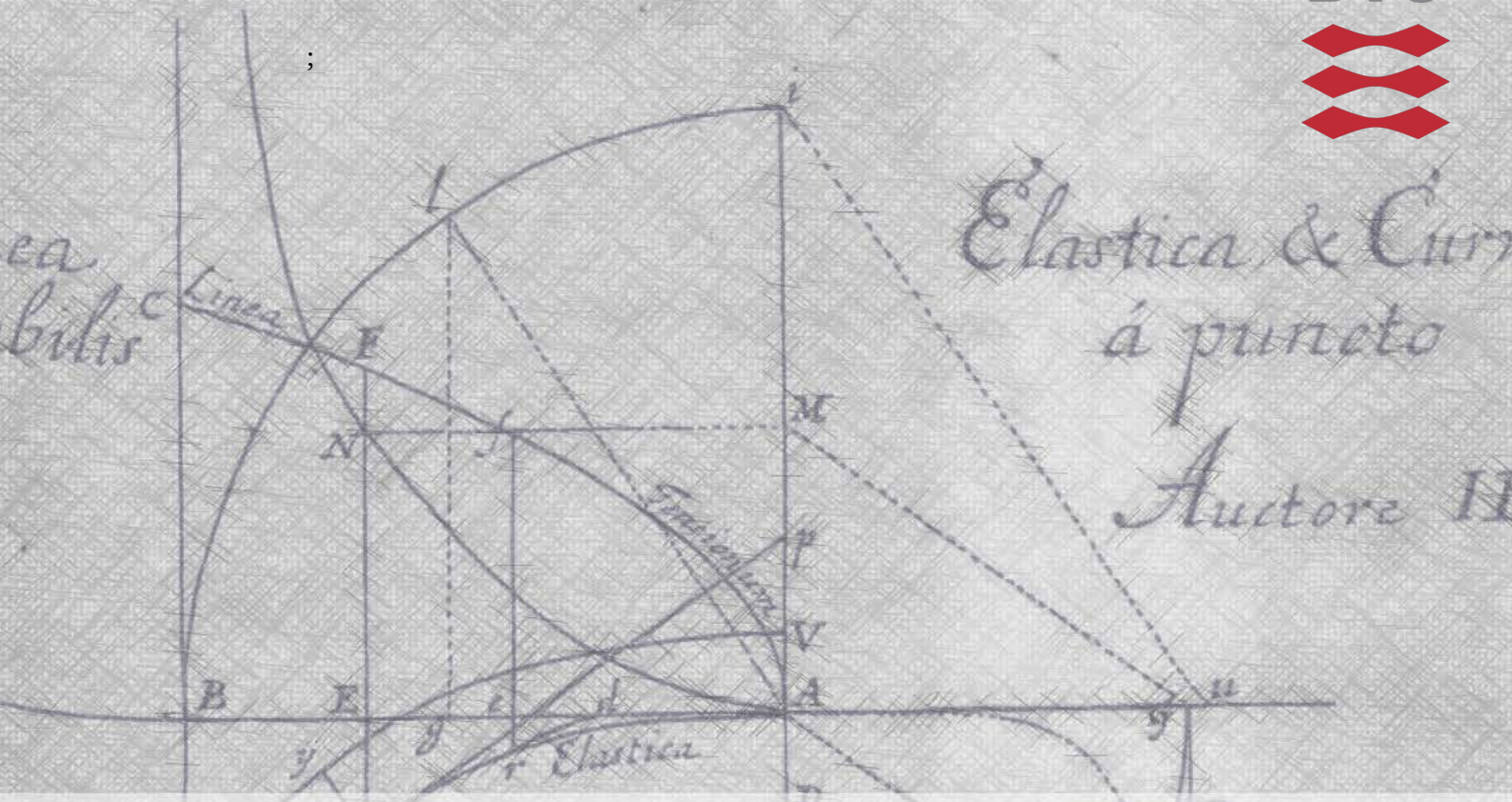
Surface design and rationalization for robotic hot-blade cutting

Fisker, Ann-Sofie

Publication date:
2019

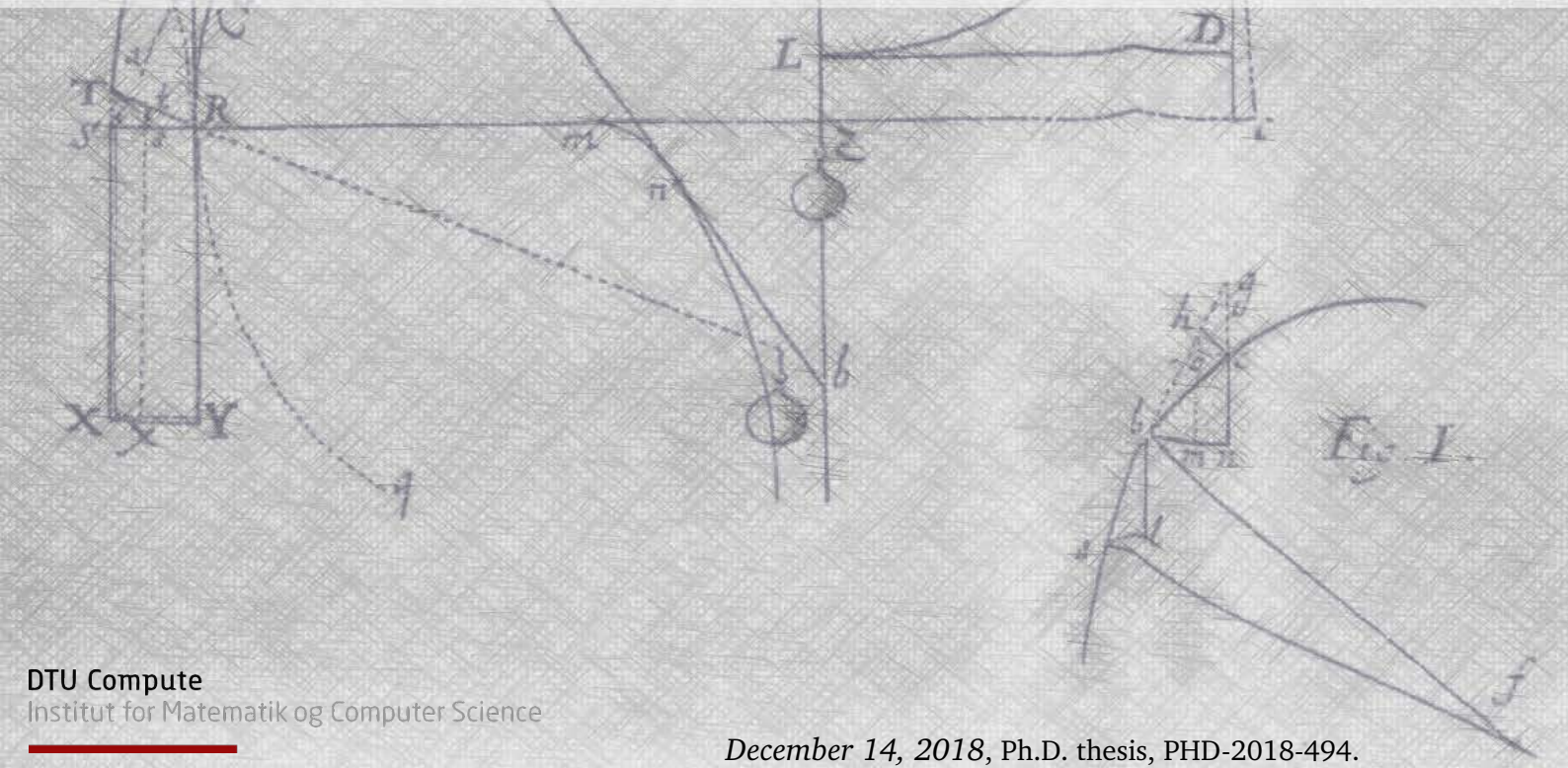
Document Version
Publisher's PDF, also known as Version of record

;



Surface design and rationalization for robotic hot-blade cutting

Ann-Sofie Fisker



DTU Compute
Institut for Matematik og Computer Science

December 14, 2018, Ph.D. thesis, PHD-2018-494.

Technical University of Denmark

Department of Applied Mathematics and Computer Science
Section for Mathematics
Geometry Group

Geometry and Design

Surface design and rationalization for robotic hot-blade cutting

Ann-Sofie Fisker

Supervisors David Brander, Jens Gravesen, and Jakob Andreas Bærentzen

December 14, 2018

Ann-Sofie Fisker

Surface design and rationalization for robotic hot-blade cutting

Geometry and Design, December 14, 2018
Supervisors: David Brander, Jens Gravesen, and Jakob Andreas Bærentzen

Technical University of Denmark
Geometry Group
Section for Mathematics
Department of Applied Mathematics and Computer Science
Anker Engelunds Vej 1
2800 Lyngby

The picture on the front page is from Bernoulli's publication of the elastic curve.

Contents

Abstract

In this thesis, we are motivated by robotic hot-blade cutting. Hot-blade cutting is a technology where a robot moves a heated flexible rod through a block of expanded polystyrene (EPS). By controlling the endpoints and end tangents, the robot controls the shape of the rod. The outcome of the cutting procedure is a cut in the EPS that can be used as a mold for concrete casting. Because the rod takes the shape of an elastic curve, we can cut curved and interesting designs that otherwise would be expensive to fabricate.

To fully utilize hot-blade cutting, architects and designers need design tools for defining the shape of the rod, i.e., computational tools for designing elastic curves, elastic splines, and surfaces foliated by elastic splines in a CAD (Computer Aided Design) environment. In this thesis, we present algorithms for accommodating this need. First, we introduce algorithms for solving the boundary value problem. By using these methods, one is able to design a single elastic curve via specification of endpoints, end tangents, and curve length. We compare the algorithms and describe their advantages and disadvantages. One of the disadvantages of solving the boundary value problem is the non-uniqueness of the solution. Consequently, the methods lack shape control. We, therefore, propose to design the elastic curves through a cubic curve interface. We present a projection tool that projects a cubic Bézier curve to a cubic curve visually close to an elastic curve. The projection keeps the endpoints and end tangent angles fixed. Along with the design of a single elastic curve, we also introduce algorithms for designing elastic splines, i.e., concatenated elastic curve segments. Among others, we present a data-driven tool for interactively designing an elastic spline. Finally, we extend the spline algorithms to the design of surfaces foliated by elastic splines.

Resumé (Abstract)

Vi er i denne afhandling motiveret af robotic hot-blade cutting. Hot-blade cutting er en teknologi, hvor en robot skærer en varm fleksibel stang igennem en blok af flamingo. Ved kontrol af endepunkter og endetangenter former robotten stangen. Resultatet af skæringsprocessen er et snit i flamingoen, der kan benyttes som beton støbeform. Fordi stangen tager form af en elastisk kurve, kan vi opnå kurvede og interessante designs, der ellers er dyre at fabrikere.

For at arkitekter og designere kan udnytte teknologien, har de brug for designredskaber for kontrol af stangens form, dvs. computer redskaber for at designe elastiske kurver, elastiske splines og flader folieret med elastiske splines i et CAD (Computer Aided Design) miljø. I denne afhandling præsenterer vi algoritmer der imødekommer dette behov. Vi vil først introducere algoritmer, der løser randværdi problemet. Ved at benytte disse metoder er det muligt at designe en enkel elastisk kurve gennem specifikation af endepunkter, endetangenter og kurvelængde. Vi sammenligner algoritmerne samt beskriver deres fordele og ulemper. En ulempe ved disse metoder er at løsningskurven til randværdi problemet ikke er entydigt bestemt, og følgeligt mangler vi kontrol af kurvens form. Vi foreslår derfor at designe de elastiske kurver via kubiske kurver. Vi introducerer et projektionsredskab, der projicerer en simpel kubisk Bézier kurve til en kubisk kurve, der er visuelt tæt på en elastisk kurve. Projektionen bevarer endepunkter og ende tangentvinkler. Udover design af en enkel elastisk kurve, vil vi også præsentere algoritmer for design af elastiske splines, dvs. kurver der stykkevis består af elastiske kurver. Vi vil blandt andet introducere en datadrevet metode for interaktivt design af en elastisk spline. Endeligt vil vi udnytte algoritmerne til design af flader folieret med elastiske splines.

Preface

The purpose of this thesis is to explain the Ph.D. project undertaken and the achieved results. The Ph.D. project, *Surface design and rationalization for robotic hot-blade cutting*, was conducted from December 2015 – December 2018 by Ann-Sofie Fisker with the supervisors Associate Professor David Brander (main supervisor), Associate Professor Jens Gravesen, and Associate Professor J. Andreas Bærentzen. The project is part of the Digital Factory project – a collaboration between DTU Compute, GXN[1], and Odico ApS[2], partly funded by Innovation Fund Denmark. During the project period three papers were written:

- Designing for hot-blade cutting: geometric approaches for high-speed manufacturing of doubly-curved architectural surfaces (D. Brander, J. A. Bærentzen, K. Clausen, A.-S. Fisker, J. Gravesen, M. Lund, T. Nørbjerg, K. Steenstrup, and A. Søndergaard). Proceedings of Advances in Architectural Geometry, Zurich 2016 [Bra+16].

- Bézier curves that are close to elastica (D. Brander, J. A. Bærentzen, A.-S. Fisker, and J. Gravesen). Computer-Aided Design, 2018 [Bra+18a].

- Designing interactively with elastic splines (D. Brander, J. A. Bærentzen, A.-S. Fisker, and J. Gravesen). Computer Aided Geometric Design, 2018 [Bra+18b].

The outcome and results of the project were presented at five conferences: Advances in Architectural Geometry (AAG) Zurich 2016, International Geometry Summit (IGS) Berlin 2016, SIAM Conference on Industrial and Applied Geometry Pittsburgh 2017, International Conference on Geometric Processing (GMP) Aachen 2018, and Advances in Architectural Geometry (AAG) Gothenburg 2018. In addition to the presentations at the conferences, an internal summer school on robotic hot-blade cutting was also organized at DTU June 2017. At the summer school the software and algorithms, developed in the above mentioned papers, were put to the test by Ph.D. students from DTU and abroad.

[1] Danish architectural practice.
[2] Danish robotics company.

Software

All algorithms described in this thesis are implemented in one or more programming languages. We have made the choice that most of the code should be useful in the sense that it can be called from Rhinoceros 5: a commercial 3-D graphics and CAD software system. Rhinoceros 5 was chosen because Rhinoceros is widely used in the design community.

In Chapter 4 we describe a projection tool, which we have implemented as a Python script and a MATLAB 2016b program. An early variation of the tool can also be found online as a JavaScript on `http://geometry.compute.dtu.dk/Bezierscript.html` (accessed October 8, 2018). The Python script can be called from the Python editor inside Rhinoceros 5. In addition to the projection tool, we have also developed a DLL file with the following features:

- computation of a discrete elastic curve that solves a boundary value problem,
- evaluation of an analytical elastic curve,
- a rough approximation of any curve with an analytical elastic curve, and
- a tool for interactively designing elastic splines with a database.

The DLL file is useful because you can call it from the Python editor inside Rhinoceros or include it in a Grasshopper script. Furthermore, the DLL file was written in C++, which has a better runtime compared to other higher level programming languages. In Chapter 5 we present a data-driven tool for interactively designing surfaces foliated by elastic splines. We have implemented this tool as a MATLAB 2016b program. The output of this design tool is a code used by RobotStudio, the software program used for controlling the robots. The code takes the format of a text file that specifies the endpoints and end tangents of the rod. This way we have implemented the complete workflow from design to production. The DLL file (with some of the features) and the MATLAB code can be downloaded at `http://github.com/TheCurveBender/DigitalFactory` (accessed October 15, 2018).

Acknowledgments

There are many whom I should acknowledge for their help, contribution, and support with the completion of the project. I could not have done it alone. To begin, I would like to thank my supervisors, David Brander, Jens Gravesen, and Andreas Bærentzen, for hiring me for the Ph.D. position. I am grateful for having the opportunity to grow from being a student in pure mathematics to becoming a researcher in applied geometry. I would also like to thank my supervisors for having invested their time in the project and my development as an early-stage researcher. I want to thank David Brander for the weekly meetings, where he has shown great interest and patience with the progress of the project and my research profile. It has been inspiring to work with him and his admirable approach to research. Along with David, Jens Gravesen has contributed his extensive knowledge to scientific discussions. His attention to detail has meant that I now think more about how I do research and express myself. Andreas Bærentzen has, with his profile in computer graphics, contributed to the implementation of the algorithms. Besides his remarkable technical knowledge, he has often demonstrated a forward-looking outlook on the project and how it could be further developed. In addition to the great support of my three supervisors, I also found support outside Denmark. Special thanks to Konrad Polthier and the Mathematical Geometry Processing group for hosting me. It gave me great insight and motivation to work in a creative environment with talented Ph.D. students. I also want to express my gratitude to the Innovation Fund Denmark for having partly funded the Ph.D. and making the research possible. I want to thank the Otto Mønsted's fund for supporting my participation in conferences. I would also like to thank the industrial partners, GXN and Odico ApS, for their highly valuable feedback on our algorithms and research.

Finally, I should mention the people who have given me substantial emotional support. My parents, together with my brother Thomas, have always told me not to worry about the future. They have told me that it does not matter if I fail because "*when a door closes, a window opens*." With this in mind, I have managed to persevere and keep my ambitions high. Besides my family, I want to thank my long-term friend Aysegül for supporting my (sometimes unrealistic and crazy) science dreams during our time at Allerød Gymnasium. Without these dreams, I would not have dared to aim for a Ph.D.

Lyngby, December 14, 2018

Ann-Sofie Fisker

1 Introduction

The thesis is directed to anyone with an interest in architectural geometry. Also, anyone with an interest in the applications of robotic hot-blade cutting might find some of the chapters useful. Most of the results presented in this thesis are algorithms developed on the basis of experiments and study of some relatively simple curves. Thus most of the thesis is self-contained, but a basic understanding of differential geometry is recommended. The thesis consists of the following six chapters:

- In Chapter 1 (this chapter) we explain the motivation behind the project and introduce the problem that we consider. In particular, we describe robotic hot-blade cutting, its advantages and limitations.

- The first section of Chapter 2 serves as a theoretical foundation for Chapters 3 to 5. We introduce the elastic curve and the λ-residual of a planar curve, which we later use in our experimental studies. In Chapter 2, we also consider two algorithms for designing a single elastic curve. Both algorithms solve the boundary value problem and we consider the limitations and advantages of these methods in a comparative framework.

- In Chapter 3 we present the experimental results from the paper "Bézier curves that are close to elastica" [Bra+18a]. We introduce cubic Bézier curves and explain different criteria that make them visually close to elastic curves.

- In Chapter 4 we present two projection tools from [Bra+18a]. The tools project a cubic Bézier curve to an elastica-like cubic Bézier curve.

- In Chapter 5 we explain methods for designing C^1 and C^2 elastic splines via interpolation. In this chapter we also introduce the results from the article "Designing interactively with elastic splines" [Bra+18b]. We present a data-driven tool for designing interactively with elastic splines. Finally, we generalize the spline tools to algorithms for designing surfaces swept by elastic splines.

- In Chapter 6 we conclude and suggest topics for future work.

In all chapters, we use pictures to illustrate the algorithms, results, and technology. Most of the pictures are modified versions of the figures in the published articles, all included with the permission of the publishers.

Results that appear only in this thesis

Many of the results in this thesis are published in the following articles: [Bra+16], [Bra+18a], and [Bra+18b]. The results that appear only in this thesis are:

- the design of an elastic curve with the boundary value problem in Section 2.2.2,
- the comparison of the boundary value solvers in Section 2.2.3,
- the mathematical analysis of the λ-residual and ordinary differential equation in Section 3.3,
- the study of the four-dimensional region of cubic Bézier curves with small λ-residual in Section 3.4,
- the design of elastic splines via interpolation in Section 5.1,
- the surface tool based on the elastic splines obtained via interpolation in Section 5.4.1, and
- the consideration of production constraints and the rationalization of designs with straight lines in Section 5.4.3.

1.1 Robotic hot-blade cutting

Robotic hot-blade cutting is a novel technology based on a very simple idea. The technology consists of a multi-robotic cell that controls a flexible heated rod that is swept through a block of polystyrene foam, see Figure 1.1. The rod is bent to different shapes but kept in a plane by the robot. As the rod is heated and swept through the foam, the material melts and evaporates. This way one obtains a cut in the foam block, which can later be used as a mold for concrete casting. Alternatively, the foam design can be coated and used as a design piece itself. Robotic hot-blade cutting is a price competitive solution for creating molds for curved designs. The material is cheap and the cutting procedure is fast. Without hot-blade cutting, curved designs are usually cast by handmade molds, making them very expensive. Using robotic hot-blade cutting, architects and designers can realize new ideas for curved building designs, giving new breath to the urban landscape.

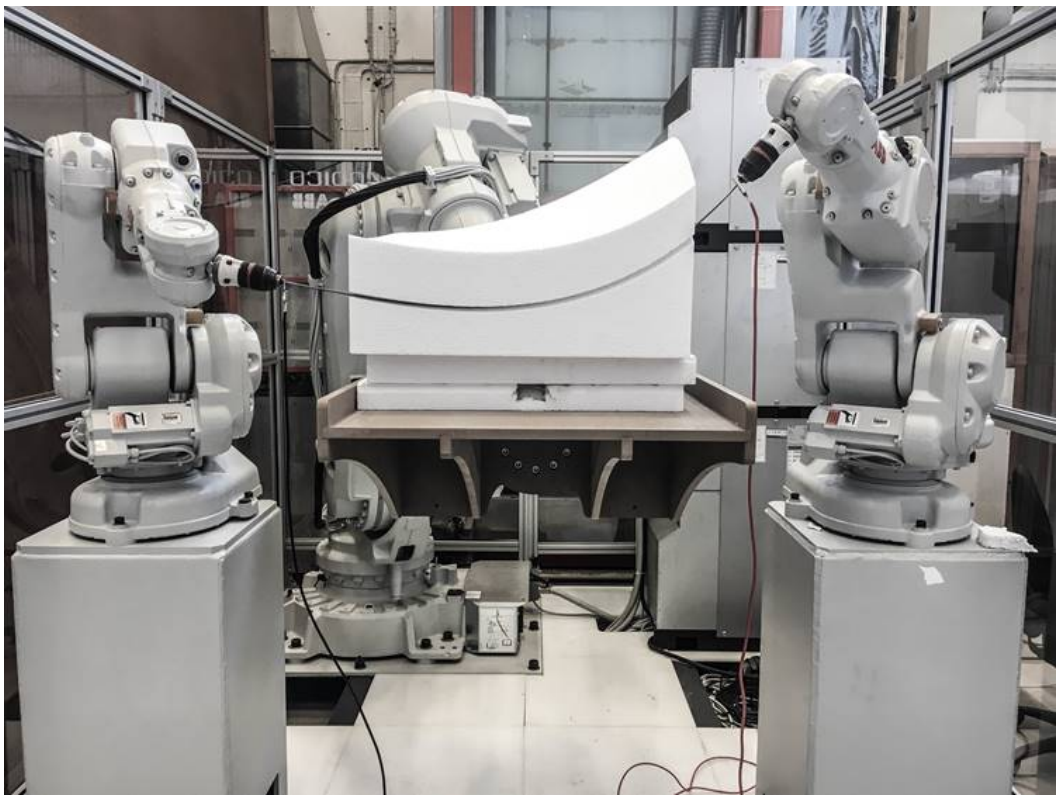


Fig. 1.1.: The robotic hot-blade cell and a fabricated design.

Motivated by curved building designs and the inexpensive and fast hot-blade technology we consider the problem of developing design tools that fully utilize hot-blade cutting. To obtain these tools we look at the shape of the rod that is swept through the foam block: the *elastica*. In the next chapter, we study the mathematics behind the elastica, also called elastic curve. Later, in the subsequent chapters, we present our algorithms for designing with elastic curves, elastic splines, and surfaces foliated by elastic splines.

1.2 Production constraints

When developing design tools for robotic hot-blade cutting a consideration of the strengths and limitations of hot-blade cutting is essential. Robotic hot-blade cutting has several strengths: the cutting procedure is fast, and because the rod is bent you can obtain designs that have positive and/or negative Gaussian curvature. Designs with positive Gaussian curvature are not achievable using the related, more widely-known, hot-*wire* technology. Hot-wire cutting is similar to hot-blade cutting but the shape of the rod is restricted to a line. With hot-wire cutting you obtain designs of ruled surfaces, which never have positive Gaussian curvature. To obtain positive Gaussian curvature an alternative subtractive manufacturing technology to hot-blade cutting is CNC milling (CNC is an acronym for Computer Numerical Control). CNC milling consists of a robot that mills a design by removing all unwanted material. With this technology, you can in principle obtain all types of designs, but compared to the hot-blade technology CNC milling is more time expensive and produces a grooved surface texture. Therefore, regarding speed and design freedom, the hot-blade cutting technology is an advantageous solution.

Despite all of the advantages, hot-blade cutting also has some noteworthy production constraints. At an internal summer school, June 2017, we tested the hot-blade technology with our industrial partners. A large curved foam wall was made. The wall clearly exhibited some unwanted features. Most notable were some small bumps cut into the foam. At these bumps, during the cutting process, the rod became unstable and started to wobble. There are several unstable configurations of the rod, but in most cases, these configurations happened when the rod swept through a line. In Chapter 5 we consider how to rationalize (fabricate) designs that contain lines. Besides an unstable rod there are other production constraints related to hot-blade cutting that are worth mentioning:

- we cannot bend the rod too much. If the elastic curve has a point with large curvature, the rod becomes plastically deformed, see Figure 1.3,

- the robotic arms must not collide with the workspace,

- if we rotate the rod, so different parts of the rod move at different speeds, then we might cause too much melting at one end of the rod, and

- if we use a flat blade, instead of a tube-shaped rod, we obtain foliations close to geodesic foliations [BG17].

These constraints might give us a reason for avoiding the hot-blade technology, but it should be said that in many cases these constraints are satisfied or we can find a workaround strategy. Furthermore, if these production constraints are satisfied the hot-blade cutting technology does a good job of replicating the CAD design. At a DTU summer school, some of the physically produced designs were scanned and compared to the original CAD models. The comparison suggested that we can expect a deviation of about 0.5 mm (for a design cut with a rod of length 800mm), see Figure 1.2.

1.3 Related research

The Digital Factory project is an extension of the Bladerunner project: another industrial research project at DTU Compute. In the Bladerunner project, two Ph.D. students worked on developing tools for approximating any CAD surface with elastic curves or ruled strips. Having these tools one is able to rationalize (segment and approximate) designs for respectively the hot-blade and hot-wire technologies. One of the algorithms from the Bladerunner project will also be used in in this thesis. More precisely, we use the elastica approximation algorithm presented in [Bra+17], and we study a residual related to this algorithm. In this Ph.D. project, we only consider the hot-blade cutting technology, but algorithms for the related technologies, hot-wire and CNC milling, were also developed in the Bladerunner project: e.g., [Ste+16] describes how to obtain a pre-processed design with hot-wire cutting before applying CNC milling. First, the algorithm approximates a given CAD design by ruled strips that are then cut with a hot-wire robot. After the hot-wire cutting, CNC milling is then applied to remove the remaining unwanted material. The approximation with ruled strips/surfaces has also been considered by others in the literature [HS98; CP99; FP10; Flö+12]. Contrary to these articles and the Bladerunner project, we will in this project not consider the problem of approximating with ruled surfaces or elastic curves. Instead, we will consider the problem of *designing* with elastic curves and obtain already production-ready designs. This approach has the following advantages: first we reduce the time spent on rationalizing a given design, and secondly, we are always guaranteed that what we design can be produced (this is not always the case when we approximate a surface by elastic curves or ruled strips).

From the production point of view, other technologies for creating curved designs or molds (for concrete casting) exist and have been considered by many in the literature, e.g., [Rus+16; Hac+14; Rau+12]. In [Rus+16] the authors discuss a method for obtaining curved designs by using spatial wire cutting. The technology consists of two robotic arms that sweep a rod through a block of foam. This is similar to the hot-blade technology, but unlike hot-blade cutting the heated rod is no longer kept in a plane. Thus the rod takes the shape of a spatial curve that is obtained by using the resistance of the processed material. Also, additive manufacturing of molds has been considered in the literature. In [Hac+14] a mobile robot extrudes a mesh structure: a mold that is then filled with concrete. Using this method, the authors obtain a curved prototype. Another method for obtaining a curved mold is to use a membrane mold [Rau+12]. A membrane mold is defined by a set of pistons and a stiff membrane that interpolates the points given by the pistons. Finally, we mention that people have employed 3-D printing to print houses and villas. The limitations and potential of 3-D printing in the construction industry are discussed in the review [Wu+16].

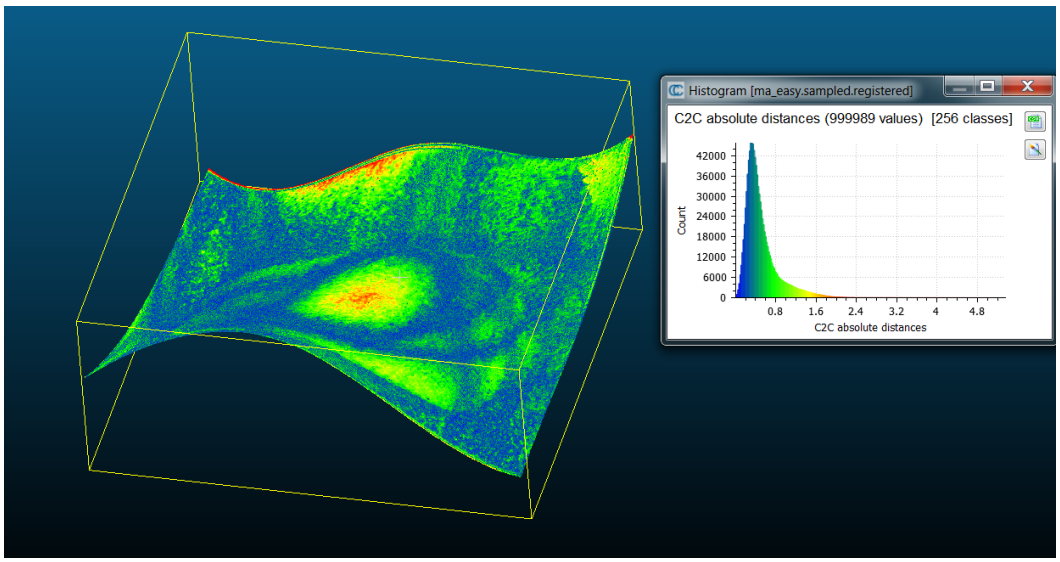


Fig. 1.2.: A scan of a design made with the hot-blade robot. We measure the distance (nearest neighbor) to the CAD model. The software program is CloudCompare [Clo17].

Fig. 1.3.: This configuration has places with high curvature and, consequently, the rod is being plastically deformed.

2 Design of elastic curves

In 1691 James Bernoulli posed a problem describing a new class of curves: the elastic curves. This class of curves was later characterized by Leonhard Euler in the work "De curvis elasticis" [Eul44]. After Euler's work in 1744, elastic curves have continued capturing the interest of mathematicians. Much later in 1892, Augustus Edward Hough Love wrote a treatise of elasticity [Lov34] and in 1906 Max Born considered the stability of elastic curves in his thesis [Bor06]. A plausible reason for this long-lasting interest is that elastic curves have many descriptions and applications in several industries. Today elastic curves are still studied as they describe the shape of a flexible rod in new technologies such as the hot-blade technology, curved wood design [BG17], gridshells [SO18], and even furniture design, see Figure 2.1. For more details on the history of the elastica, we refer to [Lev08].

Fig. 2.1.: Lamp designed by Arvid Söderholm and Ann-Sofie Fisker at AAG 2018, workshop: "Design and fabrication of bending-active structures with controlled shapes: the arc lamp".

In this chapter, we first introduce the planar elastic curves and a residual that we later use to measure the visual closeness of a cubic Bézier curve to an elastic curve. In the second section, we consider how to design a single elastic curve. We describe two methods for designing elastic curves using the boundary value problem.

2.1 The elastic curve

Before computers, curves (used for ship hulls, cars, etc.) were commonly designed by means of a set of template curves, see Figure 2.3, or by using a flexible rod, Figure 2.4 [Far02]. The flexible rod, a *spline*, was placed on a plane and kept fixed with hooked weights called ducks. Between the weights, the curve took the shape of a fair curve, the elastic curve, also called elastica or Euler's elastica. Because of the fairness, these curves are useful for a lot of designs. The reason why elastic curves are fair follows from the fact that elastic curves minimize the bending energy $\frac{1}{2}\int_0^L \kappa(s)^2 \,\mathrm{d}s$.

Definition 1 (The boundary value problem)**.** *Given fixed endpoints, end tangent angles, and curve length L the elastic curve is an arc length parameterized C^2 curve γ that minimizes the energy*

$$\frac{1}{2}\int_0^L \kappa(s)^2 \,\mathrm{d}s\,, \tag{2.1}$$

where κ denotes the curvature of γ.

In Section 2.2 we obtain numerical solutions to this problem, and we will refer to this problem as the boundary value problem. We solve this problem numerically because it does not have a closed analytic solution.

Even though we do not have an analytic solution to a given boundary value problem, we can obtain the parameterization of all elastic curves by applying the method of Lagrange multipliers [NW06]. Using this method one can show that elastic curves (up to rotation and scaling of the ambient space with non-constant curvature) satisfy $\ddot{\theta} = -\sin\theta$, where θ is the tangent angle function. From this equation we obtain the analytical description of a *basic* elastic curve as

$$\xi_k(s) = (2E(s,k) - s, 2k(1 - \mathrm{cn}(s,k)))\,,$$

where cn is one of the the basic Jacobi elliptic functions, E is the incomplete elliptic integral of the second kind, and $k \in [0,\infty)$. The derivation of this parameterization can be found in [Nør16]. For more information on the elliptic functions and the domain of k, we refer the reader to Appendix A. Because ξ_k, $k \in [0,\infty)$ only gives us elastic curves up to rotation, translation, and scaling of the ambient space, we need in total seven real parameters to describe a general elastic curve segment:

- 1 parameter $k \in [0,\infty)$ that gives us the basic elastic curve ξ_k,
- 2 parameters that specify the start and endpoint of the segment,
- 4 parameters for the translation in $\mathbb{R}^2$, rotation, and scale

= in total 7 parameters.

In Figure 2.2 we plot ξ_k for different choices of k. From the plots, one immediately sees that there are two different types of elastic curves. One type with inflectional points and one without. Also, because of the periodicity of the elliptic functions the elastic curves are translational periodic.

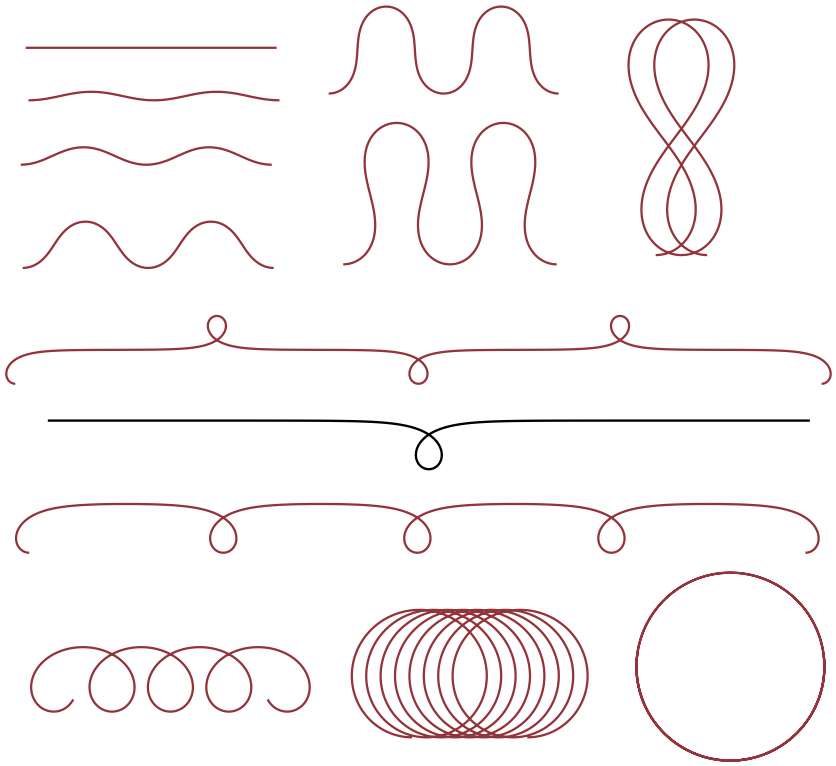


Fig. 2.2.: Elastic curves ξ_k for different values of k. The black curve is ξ_1, the straight line is ξ_0, and the circle is ξ_k for $k \to \infty$.

2.1.1 Approximation algorithm

In [Bra+17] the authors present an algorithm for approximating a planar curve by an elastic curve. The algorithm applies an optimization routine that minimizes the L^2 distance between the input curve and an approximating elastic curve given by the seven parameters. To obtain the initial guess of an approximating elastic curve, the algorithm determines the constants $(\lambda_1, \lambda_2, \alpha) \in \mathbb{R}^3$ such that the following equation, satisfied by an elastic curve, is satisfied as accurately as possible in a least squares sense

$$\kappa = \lambda_2 x - \lambda_1 y + \alpha\,, \tag{2.2}$$

where κ denotes the curvature of the input curve γ and x, y are respectively the first and second coordinate of γ.

Fig. 2.3.: Template curves called French curves. Photo: Seth Newsome.

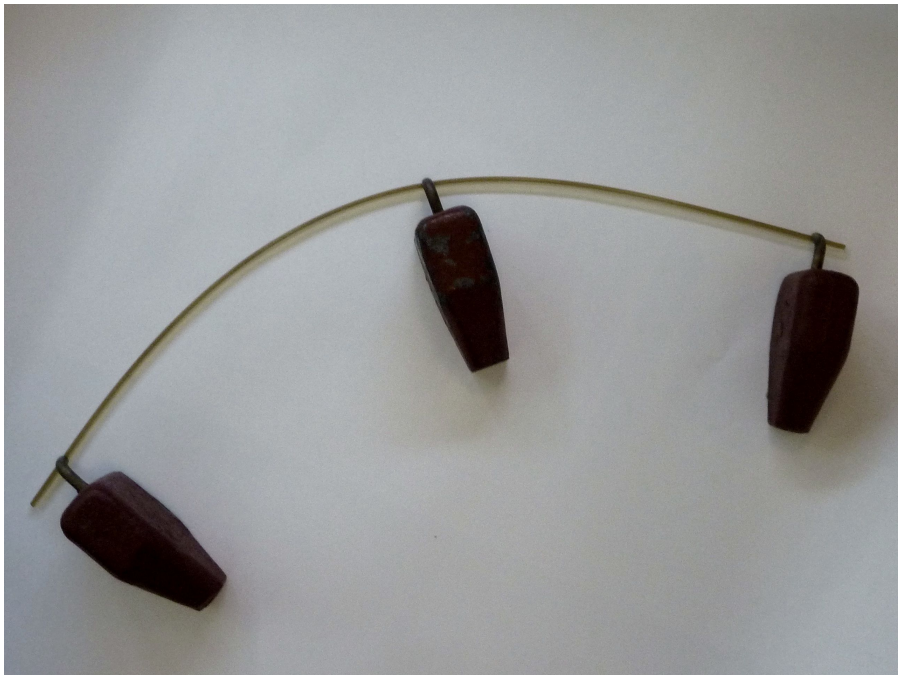


Fig. 2.4.: A physical spline on a table held in place by ducks.

The values λ_1, λ_2, and α are obtained by solving a linear system

$$\begin{pmatrix} \int_0^L y^2 \,\mathrm{d}s & -\int_0^L xy \,\mathrm{d}s & \int_0^L y \,\mathrm{d}s \\ -\int_0^L xy \,\mathrm{d}s & \int_0^L x^2 \,\mathrm{d}s & \int_0^L x \,\mathrm{d}s \\ -\int_0^L y \,\mathrm{d}s & \int_0^L x \,\mathrm{d}s & \int_0^L 1 \,\mathrm{d}s \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \alpha \end{pmatrix} = \begin{pmatrix} -\int_0^L y\kappa \,\mathrm{d}s \\ \int_0^L x\kappa \,\mathrm{d}s \\ \int_0^L \kappa \,\mathrm{d}s \end{pmatrix},$$

where s is the arc length parameterization and L is the curve length of γ. To get the seven elastic parameters of the initial guess, the algorithm uses $(\lambda_1, \lambda_2, \alpha)$ and solves another equation in the least squares sense. Throughout the thesis, we will refer to this as the initial guess. For more details on the approximation and initial guess algorithm we refer the reader to [Bra+17]. We will later use (2.2) and $(\lambda_1, \lambda_2, \alpha)$ to measure the visual closeness of a curve $\gamma : [0, 1] \to \mathbb{R}^2$ to an elastic curve. To do this we introduce the scale-invariant residual $e_\lambda \in [0, 1]$ (we call it the λ-residual) of γ:

$$e_\lambda = \sqrt{\int_0^1 (\kappa + \lambda_1 y - \lambda_2 x - \alpha)^2 \frac{\mathrm{d}s}{\mathrm{d}t} \,\mathrm{d}t} \Big/ \sqrt{\int_0^1 \kappa^2 \frac{\mathrm{d}s}{\mathrm{d}t} \,\mathrm{d}t}\,. \tag{2.3}$$

2.2 Design of an elastic curve with the boundary value problem

Instead of approximating with elastic curves we will, in this thesis, consider the problem of designing with elastic curves and elastic splines in a CAD environment. The history of CAD goes back to the 1950s where the design of curves started to transition into a computational environment. Instead of designing curves with a physical spline or French curves, people began designing curves on a computer. This transformation led to numerous algorithms for emulating an elastic curve in a CAD system, e.g., [Hor83; Bru+96]. In this section, we introduce and compare two algorithms for emulating elastic curves. Using both methods we numerically solve the boundary value problem with a polynomial spline. The two methods we consider are:

1. A linear spline solution. We obtain the numerical elastic curve using a linear spline. This linear spline solution was first introduced in [Bru+96]. In [Bru+01] the authors address the convergence to an elastica as the number of segments increases.

2. A cubic spline solution. We obtain the numerical elastic curve using a cubic spline.

When we generate the code for the hot-blade robot, we need the endpoints and end tangents of the rod that go beyond the numerical elastic curve. To obtain this information we have to extend the curve length of the numerical solutions to the

length of the rod. To extend a numerical solution, we first determine the analytical description of the elastic curve by applying the approximation algorithm above. Once we have the parameterization, we utilize that the elastic curve has constant speed and extend the length through the domain.

2.2.1 Method 1: the linear spline solution

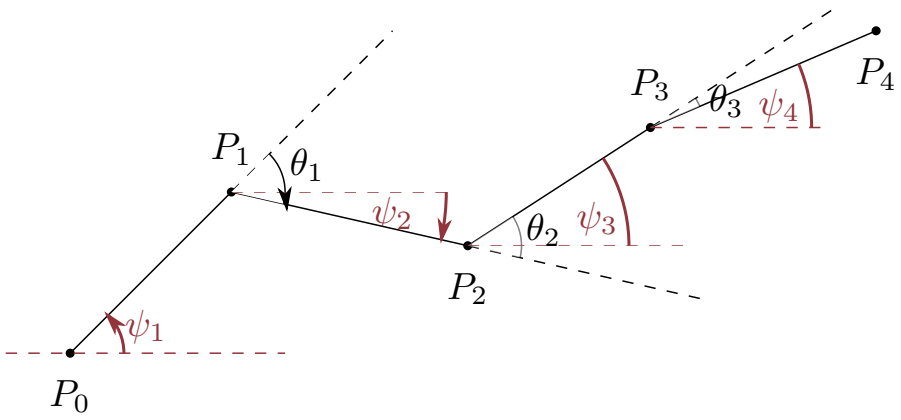


Fig. 2.5.: The linear spline model for $n = 3$ for illustration purpose only. Four line segments are not sufficient for representing a general elastic curve.

In Method 1 we represent the numerical elastic curve as a linear spline where each line segment has length l. We denote the turning angles θ_i and the angles between the x-axis and the edges ψ_i, see Figure 2.5. From the turning angles $\theta = \{\theta_i\}_{i=1}^n$ we obtain the discretization of the bending energy E_θ:

$$E_\theta = \frac{1}{2}\sum_{i=1}^{n} \theta_i^2 \,.$$

To obtain the numerical elastic curve we minimize E_θ with respect to θ and subject to fixed endpoints P_0, P_{n+1}, end tangents ψ_1, ψ_{n+1}, and length L. The constraints for P_0, ψ_1, and L are included directly in the linear spline representation. To satisfy the constraints for P_{n+1} and ψ_{n+1} we minimize E_θ subject to

$$\psi_1 + \sum_{i=1}^{n} \theta_i = \psi_{n+1} \,,$$
$$P_0 + \begin{pmatrix} \sum_{i=0}^{n} l \cos \sum_{j=0}^{i} \theta_j \\ \sum_{i=0}^{n} l \sin \sum_{j=0}^{i} \theta_j \end{pmatrix} = P_{n+1} \,,$$

where we set $\theta_0 = \psi_1$. These equations for the boundary values are highly non-linear. We can simplify the constraints by replacing the variable, turning angles $\theta = \{\theta_i\}_{i=1}^n$,

with the angles $\psi = \{\psi_i\}_{i=1}^{n+1}$. Because $\theta_i = \psi_{i+1} - \psi_i$, we get, with a change of variables, the discretized bending energy

$$E_\psi = \frac{1}{2}\sum_{i=1}^{n}(\psi_{i+1} - \psi_i)^2\,,$$

which we minimize subject to only one constraint (we include the other boundary value constraints directly in the solution),

$$P_0 + \begin{pmatrix} l\sum_{i=1}^{n+1}\cos\psi_i \\ l\sum_{i=1}^{n+1}\sin\psi_i \end{pmatrix} = P_{n+1}\,.$$

To obtain the numerical solution we apply the method of Lagrange multipliers [NW06]. That is we consider the problem of determining stationary points of the Lagrange function $\mathcal{L}(\psi, \lambda_1, \lambda_2)$ where λ_1 and λ_2 are the Lagrange multipliers. In our case we have

$$\mathcal{L}(\psi, \lambda_1, \lambda_2) = \frac{1}{2}\sum_{i=1}^{n}(\psi_{i+1} - \psi_i)^2 + \lambda_1\left(l\sum_{i=1}^{n+1}\cos\psi_i - \Delta x\right) + \lambda_2\left(l\sum_{i=1}^{n+1}\sin\psi_i - \Delta y\right)\,,$$

where $(\Delta x, \Delta y)^T = P_{n+1} - P_0$. To obtain a stationary point of $\mathcal{L}$ we apply Newton's method

$$\begin{pmatrix} \{\psi_i\} \\ \lambda_1 \\ \lambda_2 \end{pmatrix}_{j+1} = \begin{pmatrix} \{\psi_i\} \\ \lambda_1 \\ \lambda_2 \end{pmatrix}_{j} - H^{-1}(\nabla\mathcal{L}(\psi, \lambda_1, \lambda_2))\,, \tag{2.4}$$

where H is the Hessian matrix of $\mathcal{L}(\psi, \lambda_1, \lambda_2)$. We use Newton's method because it has a good convergence rate. However, in some cases it fails to converge. If $\nabla\mathcal{L}(\psi, \lambda_1, \lambda_2)$ starts increasing we use another solver based only on the gradient of E_ψ (gradient descent) with a penalty on the constraints c_1 and c_2:

$$c_1 = l\sum_{i=1}^{n+1}\cos(\psi_i) - \Delta x\,,$$
$$c_2 = l\sum_{i=1}^{n+1}\sin(\psi_i) - \Delta y\,.$$

Having established the optimization routines, we need an initial guess for the solution. One could use a line as an initial guess, $\psi_i = 0$, but this choice often leads to solutions with too many inflection points and poor performance of the optimization routines. Instead we propose to sample $\{\psi_i\}$ from a cubic Bézier curve that satisfies the boundary value constraints. We obtain the curve by fixing the endpoints and end tangent angles while adjusting the inner control points to get the correct curve length (in the next section, Method 2, we explain how we do this).

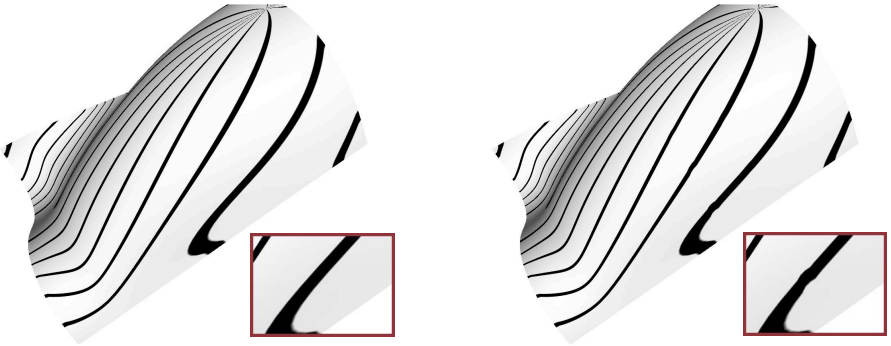


Fig. 2.7.: Two sets of curvature lines on meshes. The meshes are created by lofting a set of linear splines that solve a continuous set of boundary value problems. The right mesh was computed faster (2.6 vs. 2.1 seconds) using both Newton's method and the gradient-driven approach. Graphics: J. Andreas Bærentzen.

In Figure 2.6 we demonstrate Method 1, where we sample boundary values from three (random) curves and solve the corresponding boundary value problem. By using a combination of both the gradient based and Newton's method we get a fast solver, but if we want a foliation of curves that solve a continuous set of boundary value constraints, then the shift between the methods becomes visible in the curvature lines due to the different stopping criteria, see Figure 2.7.

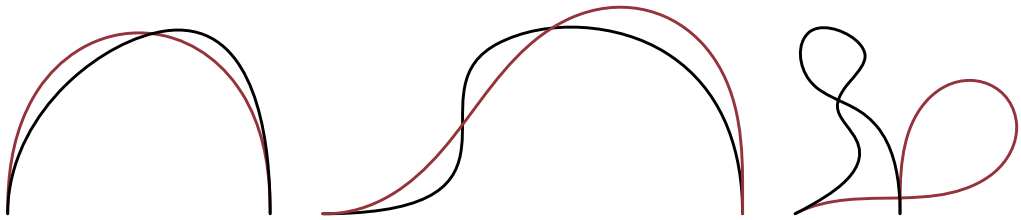


Fig. 2.6.: Numerical elastic curves (red), computed with Method 1, given by the end tangents, endpoints, and length of the black curves (chosen randomly).

2.2.2 Method 2: the cubic spline solution

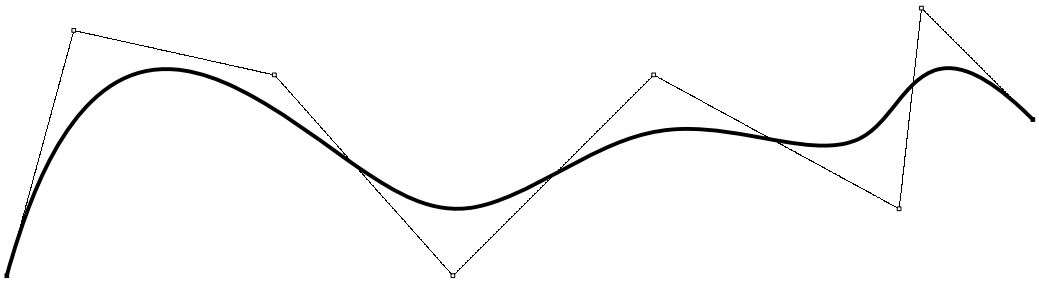


Fig. 2.8.: A cubic spline and the control polygon.

Instead of solving the boundary value problem with a linear spline we present a method that emulates the elastic curve using a cubic spline, see Figure 2.8. To

emulate the elastic curve of a boundary value problem, we use the cubic spline $\gamma_{c_1,\dots,c_n}$ given by the equation

$$\gamma_{c_1,\dots,c_n}(t) = \sum_{i=1}^{n} c_i B_{3i}(t)\,, \quad t \in [0,1]\,,$$

where $n \geq 4$ denotes the number of control points $c_i \in \mathbb{R}^2$ and $B_{3i} = B_i$ are the B-splines of order 4 [PT97]. The knot vector is a uniform knot vector τ

$$\tau = (0,0,0,0,\frac{1}{n-3},\dots,\frac{n-4}{n-3},1,1,1,1)\,.$$

We easily specify the endpoints P_1, P_2 and end tangents t_1, t_2 in terms of the variables (in this case the control points c_i):

$$\begin{aligned} P_1 &= c_1\,,\\ P_2 &= c_n\,,\\ t_1 &= \gamma'_{c_1,\dots,c_n}(0) = 3(n-3)(c_2-c_1)\,,\\ t_2 &= \gamma'_{c_1,\dots,c_n}(1) = 3(n-3)(c_n-c_{n-1})\,. \end{aligned}$$

To include the endpoints and end tangent angles in the minimization of the bending energy, we therefore introduce the parameters v_1, $v_2 > 0$ and let $c_1 = P_1$, $c_n = P_2$, $c_2 = P_1 + v_1 t_1$, and $c_{n-1} = P_2 - v_2 t_2$. This gives us the spline representation

$$\gamma_{v_1,v_2,c_3,\dots,c_{n-2}}(t) = P_1 B_1(t) + (P_1 + v_1 t_1) B_2(t) + \sum_{i=3}^{n-2} c_i B_i(t) + (P_2 - v_2 t_2) B_{n-1}(t) + P_2 B_n(t).$$

Using $\gamma_{v_1,v_2,c_3,\dots,c_{n-2}}$ we formulate the minimization problem as

$$\min_{v_1,v_2,c_3,\dots,c_{n-2}} \frac{1}{2} \int_0^1 \frac{\det\left(\gamma'_{v_1,v_2,c_3,\dots,c_{n-2}}(t)\,,\gamma''_{v_1,v_2,c_3,\dots,c_{n-2}}(t)\right)^2}{\|\gamma'_{v_1,v_2,c_3,\dots,c_{n-2}}(t)\|^5}\,\mathrm{d}t\,, \tag{2.5}$$

subject to the constraints

$$\begin{aligned} \int_0^1 \|\gamma'_{v_1,v_2,c_3,\dots,c_{n-2}}(t)\|\,\mathrm{d}t &= L\,,\\ v_1 &\geq \epsilon\,,\\ v_2 &\geq \epsilon\,, \end{aligned}$$

where $\epsilon > 0$. The energy (2.5) is the bending energy of $\gamma_{v_1,v_2,c_3,\dots,c_{n-2}}$. We see this by inserting the expression for the curvature κ into (2.1):

$$\kappa(t) = \frac{\det\left(\gamma'_{v_1,v_2,c_3,\dots,c_{n-2}}(t)\,,\gamma''_{v_1,v_2,c_3,\dots,c_{n-2}}(t)\right)}{\|\gamma'_{v_1,v_2,c_3,\dots,c_{n-2}}(t)\|^3}\,.$$

The first constraint $\int_0^1 \|\gamma'_{v_1,v_2,c_3,\ldots,c_{n-2}}(t)\|\,\mathrm{d}t = L$ is the fixed curve length and the second constraints $v_1, v_2 \geq \epsilon$ make sure that we do not move too slow at the ends of the curve (this could lead to a bad numerical solution). To find a numerical solution to the boundary value problem we apply an optimization algorithm where we discretize the energy and the length of the curve using a Gaussian quadrature rule. As an initial guess we use a feasible cubic Bézier curve. We generate a feasible cubic curve γ_v by setting $n = 4$, $\tau = (0, 0, 0, 0, 1, 1, 1, 1)$, and

$$c_1 = P_1, \quad c_2 = P_1 + vt_1, \quad c_3 = P_2 - vt_2, \quad c_4 = P_2\,,$$

for $v \in \mathbb{R}$. After this we adjust v such that the curve length of γ_v is L. We do this by solving the unconstrained minimization problem

$$\min_v \left(\int_0^1 \|\gamma'_v(t)\|\,\mathrm{d}t - L \right)^2 .$$

We can avoid loops by additionally imposing a constraint on the control polygon. When we have obtained the initial guess, we solve the minimization problem. We get the solution using an SQP (Sequential Quadratic Programming) algorithm. It is our experience that this approach requires fewer iterations compared to an interior point approach, e.g., Figure 2.11. Once we have computed the solution to the minimization problem, we perform a knot insertion and use this curve as a new initial guess to the minimization problem (2.5). The solution space is then the set of cubic splines on the updated knot vector.

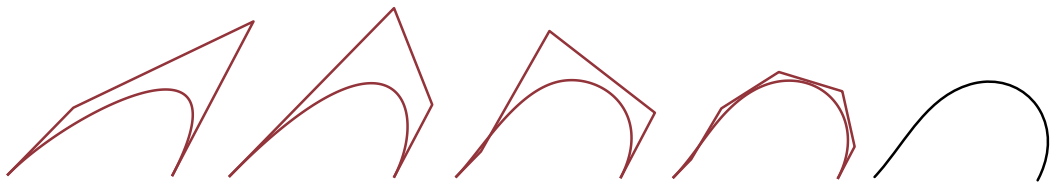


Fig. 2.9.: Method 2. We repeatedly minimize the bending energy and insert knots. The black curve is an elastic curve that solves the given boundary value problem.

Update $0 : \tau = (0, 0, 0, 0, 1, 1, 1, 1)$.

Update $1 : \tau = (0, 0, 0, 0, 0.5, 1, 1, 1, 1)$.

Update $2 : \tau = (0, 0, 0, 0, 0.25, 0.5, 0.75, 1, 1, 1, 1)$.

Etc.

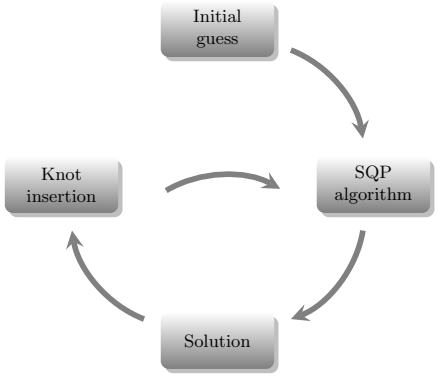


Fig. 2.10.: The algorithm of Method 2.

We repeat the procedure (minimization of the bending energy and knot insertion) until we have obtained a numerical elastic curve, see Figure 2.9 and 2.10. The knot vector is repeatedly updated such that it continues to be uniform. We determine if the curve is an numerical elastic curve by using the Hausdorff distance between the curve and updated curve. If the Hausdorff distance is small, then we terminate. To reduce the computation time one could instead of the Hausdorff distance use the much cheaper L^2 distance. Alternatively one could also use the residuals from the approximation algorithm in [Bra+17] as a stopping criterion. By using this recursive procedure we minimize the number of control points/knots needed to represent the elastica.

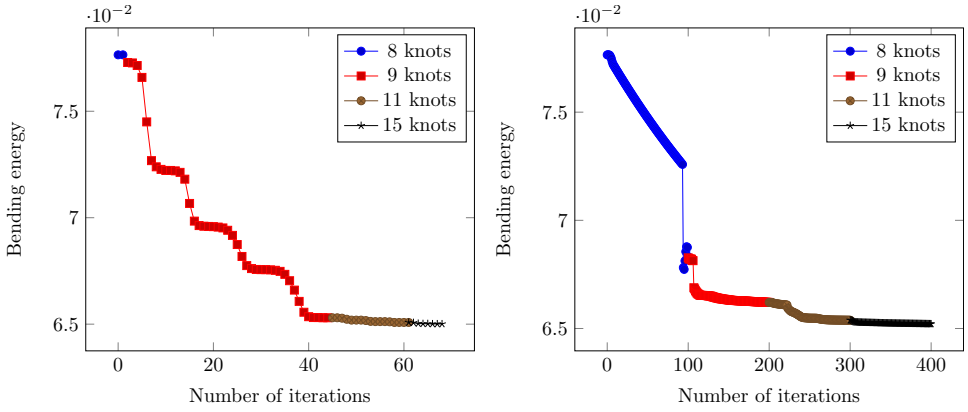


Fig. 2.11.: Application of an SQP algorithm (left) and an interior point algorithm (right) to a boundary value problem.

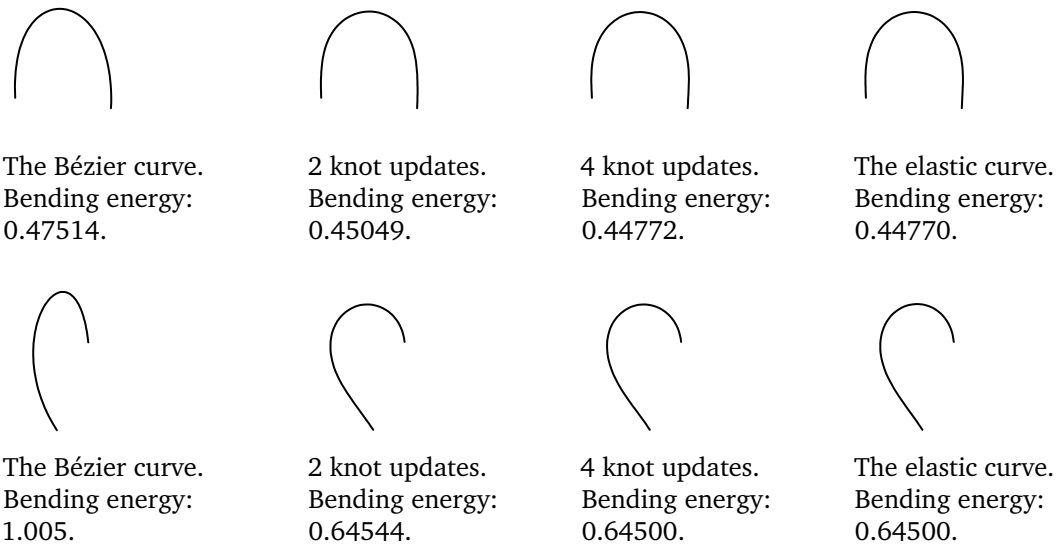
2.2.3 Comparison

We have presented two methods for designing an elastica by numerically solving the boundary value problem. To perform a comparative analysis, we address the data storage, computational time of the methods, and actual performance.

To discuss the data storage, we look at the solution spaces used in the minimization problems. The solution space of Method 1 is the set of linear splines. For a linear spline to represent an elastic curve, we generally need a large set of points compared to a higher degree polynomial spline. Hence the data storage of Method 2 is advantageous because we use a solution space of cubic splines. In Method 2 we keep the number of control points minimal by repeatedly inserting knots. In Figure 2.12 we consider two examples where we only need two knot updates to get a numerical elastica that looks identical to the actual elastica. For more examples we refer to Appendix Table B.1.

To address the computational time performance of the methods, we observe that both methods solve a non-linear optimization problem. Method 1 has a simple

energy $\frac{1}{2}\sum_{i=1}^{n}(\psi_{i+1}-\psi_i)^2$ but a highly non-linear constraint for keeping the second endpoint fixed. The position of the endpoints is important if we want to define elastic splines. If we define an elastic spline with Method 1, we cannot relax the endpoint constraint, and this has a negative impact on the computational time. With Method 2 we avoid this by including the endpoints and end tangents directly into the solution. But the objective and the length constraints are still non-linear.



The Bézier curve. Bending energy: 0.47514.

2 knot updates. Bending energy: 0.45049.

4 knot updates. Bending energy: 0.44772.

The elastic curve. Bending energy: 0.44770.

The Bézier curve. Bending energy: 1.005.

2 knot updates. Bending energy: 0.64544.

4 knot updates. Bending energy: 0.64500.

The elastic curve. Bending energy: 0.64500.

Fig. 2.12.: Two examples of numerical elastic curves computed with Method 2.

To compare the actual performance of the methods, we have implemented the algorithms as two programs in MATLAB. We used 100 gauss points in the numerical integrations, and for Method 2 we applied the SQP algorithm from the optimization toolbox. We measured the performance of the methods by extracting the boundary values of 500 random elastic curve segments with curve length 1 and at most one period. For each set of boundary values, we computed the solutions with Method 1 and 2, storing the constraint errors, runtime, L^2 and H^1 distances. In Table 2.1 we show the results. From the table, we see that Method 1 satisfies the second endpoint with a max error (Euclidean distance) of approximately 0.07, but we can expect that the error is smaller (median approximately 1e-07). Method 2 always satisfies this constraint but not the length constraint. In this case, there are few bad examples, where the optimizer fails to find the solution, and the length explodes. However, in general, the length error is small (median approximately 5e-09). In the table we also compare the runtime. Method 1, with a 250 point polyline, seems to be running 2–4 times faster compared to Method 1 with 750 points and Method 2. In the table, we also list the L^2 and H^1 distances to the elastica. The average distance is large because the methods do not always find the right elastic curve but a different elastic curve to the boundary value problem. Figure 2.13 displays two of these examples.

Based on this comparison we suggest that the reader uses the method that is more suitable for his/her application depending on the importance of boundary constraints (endpoints vs. curve length), data storage, and runtime.

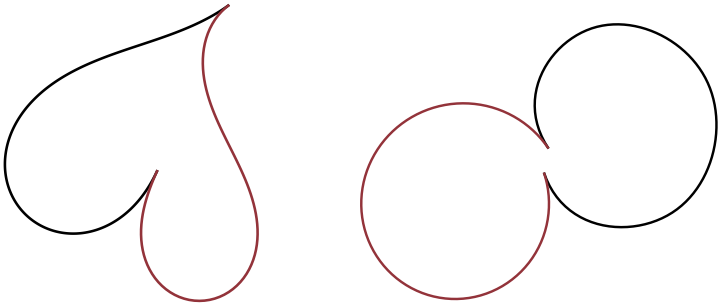


Fig. 2.13.: The red curves are the two elastic curve segments from the 500 sample and the black curves are the solutions. Left: Method 1 (L^2 distance is 0.32). Right: Method 2 (L^2 distance is 0.39).

Tab. 2.1.: We applied Method 1–2 to 500 boundary value problems and reported the performance.

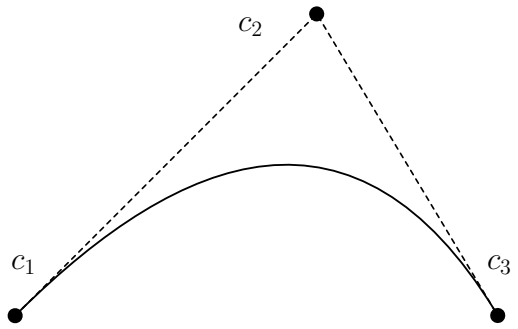
Method	1 (250 point polyline)	1 (750 points)	2 (0 knot updates)	2 (1 knot updates)	2 (2 knot updates)
Average					
2nd endpoint	5.27e-04	4.94e-04	0	0	0
Length	0	0	2.67e-05	3.75e-02	1.33e-02
Runtime (s)	1.09e-01	4.46e-01	1.38e-01	4.18e-01	4.36e-01
L^2 dist	1.88e-02	1.8e-02	3.14e-02	2.5e-02	2.20e-02
H^1	1.51e-01	1.37e-01	3.38e-01	2.18e-01	1.54e-01
Median					
2nd endpoint	1.07e-07	2.19e-08	0	0	0
Length	0	0	4.23e-09	4.50e-09	6.29e-09
Runtime (s)	9.19e-02	3.86e-01	1.01e-01	2.65e-01	3.72e-01
L^2 dist	1.78e-03	5.71e-04	2.25e-03	4.07e-04	3.82e-05
H^1	4.26e-03	1.27e-03	2.21e-02	4.42e-03	8.02e-04
Max					
2nd endpoint	6.46e-02	8.70e-02	0	0	0
Length	0	0	1.31e-02	1.22e01	6.49
Runtime (s)	5.33e-01	1.57	1.37	3.46	2.56
L^2 dist	3.14e-01	3.15e-01	3.38e-01	7.28e-01	7.28e-01
H^1	2.60	2.66	3.88	3.54	3.46

3 Cubic Bézier curves and elastic curves

In the 1950s a computer could control a milling machine. At that time most technical drawings were stored as blueprints, and consequently there was a demand for a computer compatible format for curves and surfaces. To accommodate the demand, Bézier curves were developed in the automobile industry by Pierre Bézier at Renault and Paul de Casteljau at Citröen [Far02].

3.1 Cubic Bézier curves

Bézier curves are a generalization of the fact that a parabola γ_1^2, e.g., Figure 3.1, can be determined by taking three points c_1, c_2, and c_3 and repeatedly perform a linear interpolation:



$$\gamma_1^1(t) = (1-t)c_1 + tc_2\,,$$
$$\gamma_2^1(t) = (1-t)c_2 + tc_3\,,$$
$$\gamma_1^2(t) = (1-t)\gamma_1^1(t) + t\gamma_2^1(t).$$

Fig. 3.1.: The parabola γ_1^2 given by the three points c_1, c_2, and c_3.

The procedure can be generalized such that we obtain an nth degree polynomial γ_1^n, called a Bézier curve, by using $n+1$ points $c_1, \ldots, c_{n+1}$ and computing γ_1^n with the recursive scheme for $r = 1, \ldots, n$ and $i = 1, \ldots, n+1-r$:

$$\gamma_i^r(t) = (1-t)\gamma_i^{r-1}(t) + t\gamma_{i+1}^{r-1}(t)\,,$$

where $\gamma_i^0 = c_i$. In particular we will, in this thesis, consider *cubic* Bézier curves. In Figure 3.2 we plot two examples of cubic Bézier curves with their control poly-

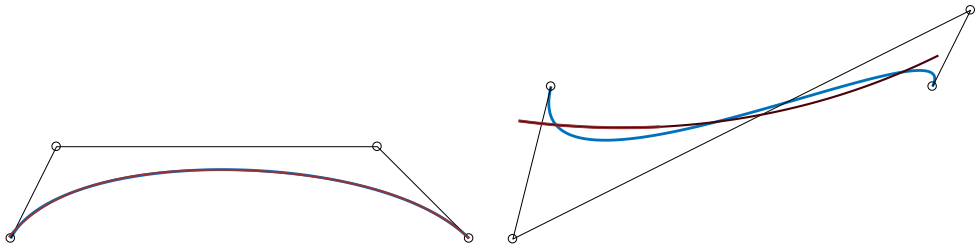
gons. Using the B-splines B_{3i} of order 4 the cubic Bézier curves are given by the parameterization

$$\gamma_{c_1,\ldots,c_4}(t) = \sum_{i=1}^{4} c_i B_{3i}(t)\,, \quad t \in [0,1]\,,$$

on the knot vector $\tau = (0,0,0,0,1,1,1,1)$. Bézier curves have many good properties. Especially they are useful as design tools for creating curves. They are easy to modify via the control points, and they are intuitive because the control polygon mimics the shape of the Bézier curve. Often cubic Bézier curves are mistaken for elastic curves. One of the reasons for this is that a cubic curve γ minimizes the following energy, for $t_1, t_2, y_1, y_2 \in \mathbb{R}, 0 \leq p$,

$$\int_{-\infty}^{\infty} \gamma''(t)^2 \,\mathrm{d}t + p((y_1 - \gamma(t_1))^2 + (y_2 - \gamma(t_2))^2)\,,$$

and if γ is parameterized by arc length, then $\kappa = \gamma''$ and the first term coincides with the bending energy [Yam13]. But, in general, cubic Bézier curves are not parameterized by arc length, and hence some curves are far from being elastic curves. In Figure 3.2 we show an example of a cubic Bézier curve not close to an elastic curve. Here we have used the approximation algorithm to obtain the approximating elastic curve.



err $= 0.0008$. err $= 0.025$.

Fig. 3.2.: Two cubic Bézier curves (blue) with their elastic curve approximation (red). The left Bézier curve is close to being an elastic curve, while the right curve deviates from its elastic curve approximation. The value err denotes the normalized L^2 distance between the Bézier and elastic curve.

3.2 Criteria for being close to an elastic curve

A large part of the Ph.D. project has consisted of determining criteria for cubic Bézier curves that make them visually close to elastic curves. The motivation for this is to obtain a design interface for creating elastic curves by restricting the designer to a subclass of cubic Bézier curves that are close to the elastic curves (we call these curves *elastica-like*). A cubic design interface has several advantages over employing a boundary value solver as a design tool. It gives us more shape control and is more robust. We also avoid minimizing the bending energy with an optimization

routine which can be time expensive and cause unexpected output. We use cubic Bézier curves because they are the lowest order Bézier curve that can be expected to represent all rod configurations. For instance, a parabola cannot represent the "$s-$curve" taken by the rod in Figure 1.3.

We can restrict the designer to elastica-like Bézier curves by providing a diagnostic tool[1] or by projecting any cubic Bézier curve into this class. The work in this and the next chapter is therefore motivated by the following questions:

1. How can we tell if a cubic Bézier curve is elastica-like?

2. How can we project a cubic Bézier curve to an elastica-like Bézier curve?

To answer these questions, we first need to clarify what we mean by "being close to an elastic curve". Given a cubic Bézier curve, we measure the deviation by computing the best elastica approximation in the L^2 sense. Using the elastica approximation, we compute the deviation as the L^2 distance between the Bézier curve and elastic curve normalized by the curve length. The normalized L^2 norm is useful. It is easy to compute but not always a suitable choice for measuring closeness to an elastic curve. In Section 3.3 we discuss other choices of norms, but for now, we adapt the criterion that a cubic Bézier curve is elastica-like if it has a normalized L^2 distance less than 0.01. We obtain the elastica approximation with the algorithm in [Bra+17]. To increase the confidence that we obtain the best possible elastic curve approximation, we apply four different optimization routines (SQP, interior-point, trust-region, and quasi-newton) [NW06]. We implemented the approximation in MATLAB by means of the optimization toolbox [MAT16].

Having defined elastica-like cubic Bézier curves, we answer our first question: "How can we tell if a cubic Bézier curve is elastica-like?". To answer the question we take an experimental approach, i.e., we generate and study a large randomized sample of cubic Bézier curves. For each curve in the sample we determine the elastica approximation and compute the L^2 distance. Having the L^2 distances, we search for different properties of the Bézier curves that make them close to an elastic curve. We use an 80,000 sample to determine criteria for being elastica-like. Because elastic curves are invariant under rotation, scale, and translation we fix the endpoints of the cubic curves at $(0,0)$ and $(1,0)$. We sample the inner control points in the two annuli with interior radii 0.25, exterior radii 1.5, and centers at $(0,0)$ and $(1,0)$. We do not consider curves with inner control points close to or far away from the endpoints because we posit that these curves do not show up in many designs. Also, we only use half of the annuli at $(1,0)$. We can obtain the remaining curves by reflecting the curves in the sample. The 80,000 sample of the inner control points is

[1] A tool that detects if a curve is elastica-like.

plotted in Figure 3.3. Here we get the sample from all combinations of the red and blue points. In the following sections, we examine different properties of the curves in this sample: the minimum angle, the λ-residual (2.3), and the geometry of the control polygon (edge lengths and angles).

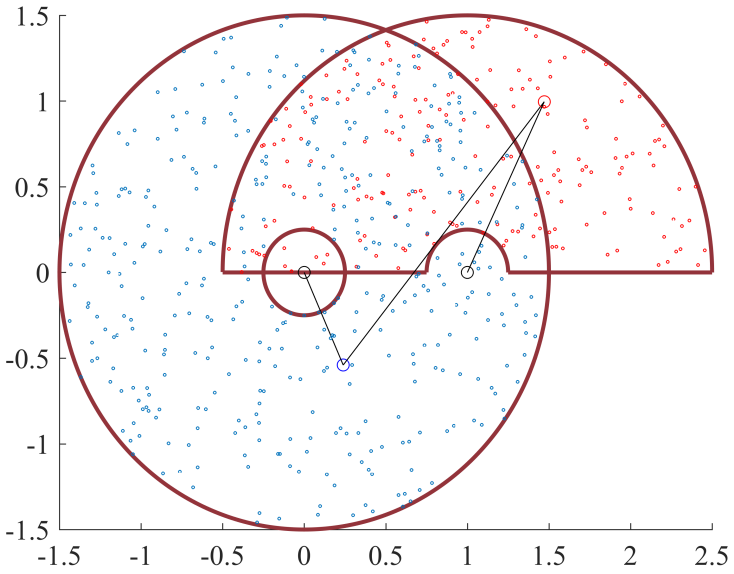


Fig. 3.3.: The sample of inner control points and one of the 80,000 control polygons. The sample is the set of all combinations of the red and blue points.

3.2.1 The minimum angle

One could expect that a cubic Bézier curve with a small angle in its control polygon is not elastica-like, e.g., the examples in Figure 3.4.

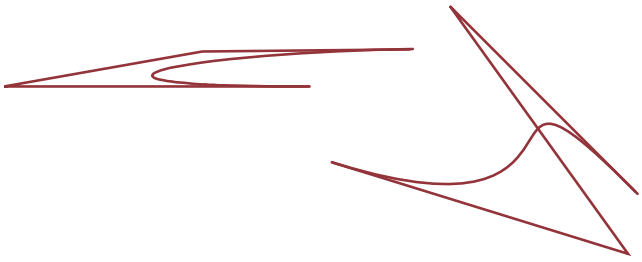


Fig. 3.4.: Bézier curves with small angles in their control polygons.

We test this expectation in our sample, where we consider the angles $\phi_1, \phi_2 \in [0, 2\pi)$ at the inner control points, see Figure 3.5. Using these angles, we compute the smallest angle as $\phi_{\min} = \min(\phi_1, \phi_2, 2\pi - \phi_1, 2\pi - \phi_2)$ and plot the L^2 distance against this quantity. From the plot in Figure 3.6, we observe that if $\phi_{\min}$ is large, then the L^2 distance is small. However, we must use the minimum angle $2\pi/3$ to get

below our 0.01 threshold for being elastica-like and the lower bound $\phi_{\min} > 2\pi/3$ is very restrictive.

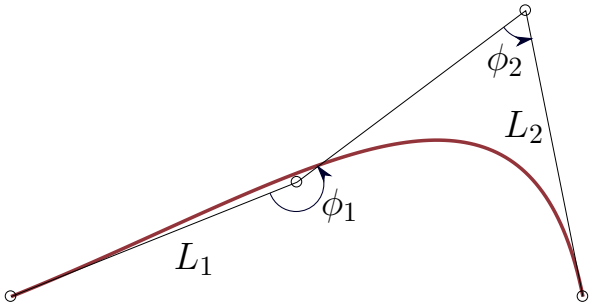


Fig. 3.5.: Bézier curve, the angles ϕ_1, ϕ_2, and outer edge lengths L_1, L_2.

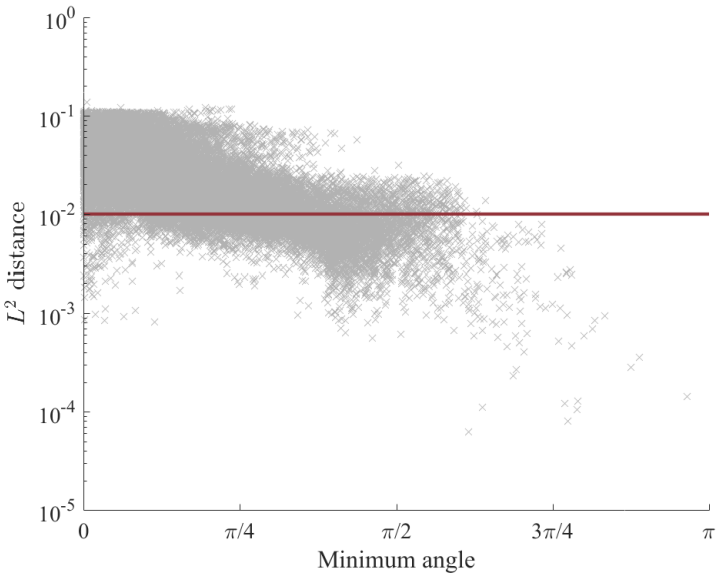


Fig. 3.6.: The normalized L^2 distance and the minimum angle.

3.2.2 The λ-residual

Instead of the minimum angle we consider the less geometrically intuitive λ-residual e_λ (2.3). Using the λ-residual, we test if we can obtain a better (less restrictive) criterion for being elastica-like. The λ-residual measures how far away the curvature κ of a curve $\gamma = (x, y)$ is from satisfying the equation of an elastica $\kappa = \lambda_2 x - \lambda_1 y + \alpha$, where $\lambda_1, \lambda_2, \alpha \in \mathbb{R}$ are the best possible choices. By plotting the L^2 distance against the λ-residual e_λ in Figure 3.7, we observe that we can get a small L^2 distance if e_λ is small without being too restrictive on the sample. We see that a maximum L^2 distance of 0.01 is obtained by almost all curves with a λ-residual less than 0.4. In Section 3.3 we explain in details why and how e_λ relates to the L^2 distance.

Even though the λ-residual is easy to compute, it is not very intuitive. Meaning that we cannot immediately tell from the shape of a cubic Bézier curve if the λ-residual is small. Also, if we want to project curves to curves with small λ-residuals, we must take a gradient-driven approach, and this approach has several issues (we explain this in the introduction of Chapter 4). However, later in Chapter 5, the λ-residual will play an essential part in our proposed algorithm for designing elastic splines via a database.

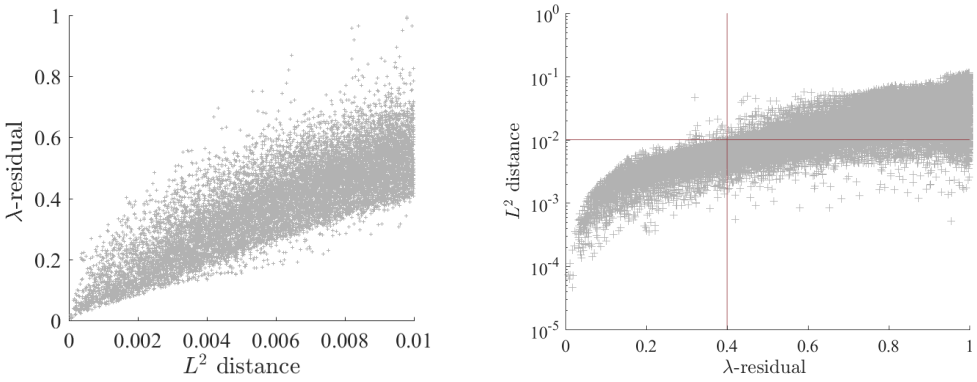


Fig. 3.7.: The λ-residual (2.3) and normalized L^2 distance.

3.2.3 The geometry of the control polygon

We now study a set of properties related to the geometry of the control polygon of the cubic curves. We search for constraints on the angles in the control polygon and the lengths of the outer edges L_1, L_2. Ideally, we want these constraints to guarantee a small L^2 distance, be not too restrictive, and geometrically intuitive (so we avoid the problems we had with the minimum angle and the λ-residual).

To determine the constraints we consider a preliminary study of the 80,000 control polygons. In this study, we classify the polygons according to the positions of the inner control points in the four quadrants. Admittedly, this is an artificial choice for categorizing the curves. However, the following results only serve as inspiration for our later study. Using the four quadrants to classify the cubic Bézier curves, we can check that there are in total sixteen configurations, but they are all represented by the six types A_1–A_2, B_1–B_2, and C_1–C_2 in Figure 3.8.

Already from Figure 3.8, we conclude that type C_1–C_2 must contain a lot of curves that are not elastica-like. These curves appear to have a more stretched shape: they have places with large curvature and an inflection point. Indeed from our 80,000 sample, we obtain that most curves in C_1–C_2 are far from being elastic curves, see the statistics in Table 3.2.

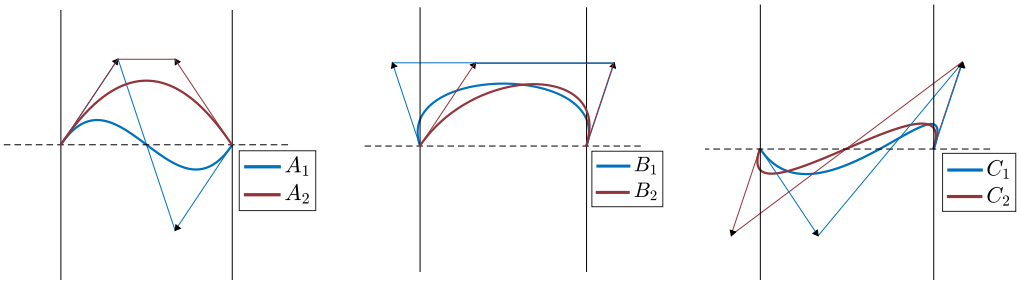


Fig. 3.8.: The six types of cubic Bézier curves based on the four quadrants. Type A_1–A_2 has both outer edges pointing inwards. Type B_1–B_2 has both edges up or down. Type C_1–C_2 has one edge up and one down, and at least one of them is pointing outwards.

Type B_1 also seems to contain curves that are bad/not elastica-like. Measuring $\max(\alpha_1, \alpha_2)$ where $\alpha_1, \alpha_2 \in [0, \pi/2]$ denote the angles from the outer edges to the vertical lines, see Figure 3.9, we observe that by restricting this we remove a lot of the bad B_1 curves, see Table 3.1. Furthermore, we obtain from our sample that the quantity Θ, given by the angle between the end tangents (for B_1: $\Theta = \alpha_1 + \alpha_2$), can be bounded to exclude the bad curves from type B_1. For type B_2 we obtain a lot of curves close to the elastic curves if α_2 is large, but not many curves with α_1 large. Hence by adding a bound on α_1 we remove some of the bad B_2 curves.

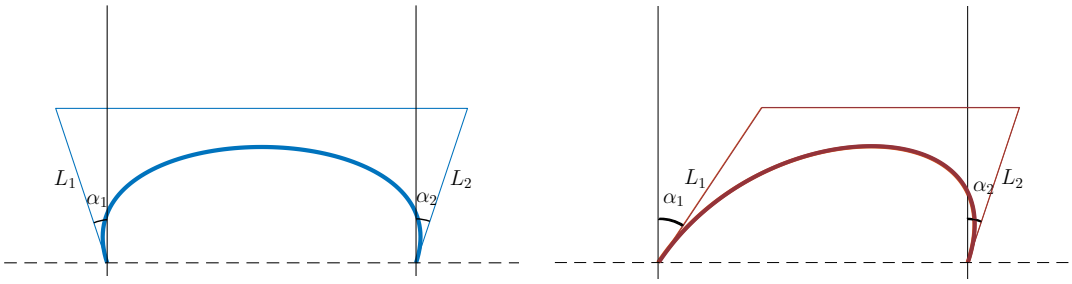


Fig. 3.9.: Type B_1–B_2 with the angles α_1 and α_2.

There are also bad type A_1–A_2 curves. If the outer edge lengths L_1 and L_2 of an A_1–A_2 curve are large, then the curve is not elastica-like. In this case we either have a self-intersecting control polygon or a curve with an inflection point, exhibiting high curvature.

In Table 3.1 and Table 3.2 we summarize the statistics for our preliminary classification of the sample. Based on this classification and statistics we now consider a larger sample of cubic Bézier curves with eight million curves. Instead of the L^2 distance we use the λ-residual e_λ to measure closeness to an elastic curve. The λ-residual can replace the L^2 distance because of the previous results in Section 3.2.2.

Tab. 3.1.: Statistic for curves of type B_1–B_2. The error is the normalized L^2 distance to the elastica approximation.

Type	#	< 0.003	$0.003 - 0.01$	$0.01 - 0.02$	> 0.02	Mean	Max
B_1	8645	295	2887	2094	3369	0.021	0.08
$B_1 : \max(\alpha_1, \alpha_2) < \pi/4$	1763	295	1231	230	7	0.0062	0.037
$B_1 : \Theta < \pi/3$	1723	285	1247	186	5	0.006	0.023
B_2	18,548	984	5859	6379	5326	0.016	0.11
$B_2 : \alpha_1 < \pi/3$	14,520	979	5634	5520	2387	0.012	0.059
$B_2 : \Theta < \pi/4$	14,542	978	4347	5019	4198	0.016	0.11

Tab. 3.2.: Statistic for all types. We have removed self-intersecting control polygons. Good means normalized L^2 distance less than 0.01.

Type	#	< 0.003	$0.003 - 0.01$	$0.01 - 0.02$	> 0.02	Good	Mean	Max
A_1	10,605	201	1988	3184	5232	20.6 %	0.023	0.098
A_2	6043	600	2485	2453	505	51.1 %	0.011	0.034
B_1	8645	295	2887	2094	3369	36.8 %	0.021	0.08
B_2	18,548	984	5859	6379	5326	36.9 %	0.016	0.11
C_1	19,885	15	952	4314	14,604	4.86 %	0.032	0.11
C_2	9310	2	221	2417	6670	2.4%	0.03	0.092
Total	73036	2097	14392	20841	35706	22.6 %	0.023	0.11

Using the λ-residual we are able to generate and study a much larger sample because the λ-residual is fast and easy to compute (contrary to applying the four optimization routines in the approximation algorithm). In the following discussion we will refer to a curve with $0.4 < e_\lambda$ as a bad/not elastica-like curve (and likewise we will call a curve with $e_\lambda \leq 0.4$ good/elastica-like).

We obtain the eight million sample by extracting four random control points in the unit disk and transforming the control polygon such that the endpoints are fixed at $(0, 0)$ and $(1, 0)$. This gives us a truly randomized sample. The 80,000 sample was also randomized but constrained by our assumption of where a designer is more likely to draw a Bézier curve. Instead of classifying the curves in the sample using the six types, we consider the general case given by the angles ϕ_1, ϕ_2 at the interior control points, the outer edge lengths L_1, L_2, and the angles β_1, β_2, see Figure 3.10. Here $\beta_1 \in [0, 2\pi)$ is the angle measured clockwise from the negative x-axis to the first outer edge and $\beta_2 \in [0, 2\pi)$ is the angle measured counter-clockwise from the positive x-axis to the last outer edge. From our preliminary study, we know that β_1 and β_2 should not be too small or large so we add the following bound $\beta_1, \beta_2 \in (\pi/3, 2\pi - \pi/3)$ as a constraint to get elastica-like curves. This bound corresponds to the exclusion of the gray region in Figure 3.10. Using this constraint, we exclude the B_1 curves with large $\max(\alpha_1, \alpha_2)$ and some of the worst curves from type C_1 and C_2.

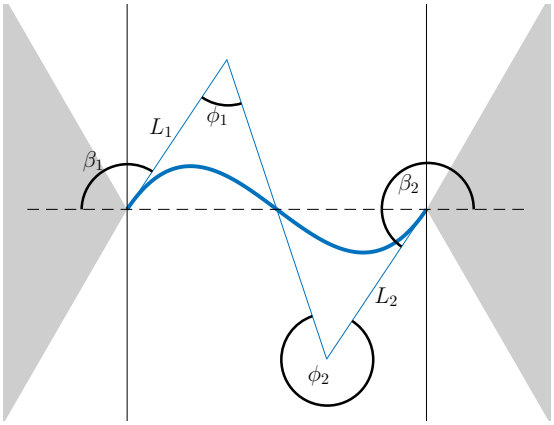


Fig. 3.10.: The control polygon of a Bézier curve with the angles $\beta_1, \beta_2, \phi_1, \phi_2$ and lengths of the outer edges L_1 and L_2.

By additionally imposing another bound $|\beta_1 - \beta_2| < 0.4\pi$, we avoid bad curves from type B_2. Summed up we impose the following two angle constraints on our sample:

$$\text{Angle Constraint 1: } \beta_1, \beta_2 \in \left(\frac{\pi}{3}, 2\pi - \frac{\pi}{3}\right).$$
$$\text{Angle Constraint 2: } |\beta_1 - \beta_2| < 0.4\pi\,.$$

With these two constraints and the constraint of not having a self-intersecting control polygon, we remove a lot of bad curves and only few elastica-like curves. In total, we have 4,339,509 curves left from our eight million sample. After having included Angle Constraints 1–2 we now add constraints on ϕ_1, ϕ_2 and L_1, L_2 such that we are guaranteed an elastica-like curve. If we plot (ϕ_1, ϕ_2) and color-code e_λ, we observe that there are bad curves with $e_\lambda \geq 0.8$ everywhere, see Figure 3.12a. However, we also find that there are a lot of elastica-like curves in the middle of the plot. Given this plot, we search for constraints on L_1 and L_2 such that we remove the bad curves in the middle of the plot. From our preliminary study, we already observed that for type A_1–A_2 we must have shorter edges, and it turns out for edges pointing outwards they should be larger. Together with a symmetry constraint for inflectional curves, we get two length constraints:

$$\text{Length Constraint 1: } L_{\min} \leq L_1, L_2 \leq L_{\max}\,,$$
$$\text{Length Constraint 2 (only for inflectional curves): } \max\left(\frac{L_1}{L_2}, \frac{L_2}{L_1}\right) < 1.3\,,$$

with

$$\begin{aligned} L_{\min} &= \max\left(0.4(1 + 6\Delta), 0.27\right)\,, \\ L_{\max} &= \max\left(1.2(1 + 5\Delta), 0.58\right)\,, \\ \Delta &= \frac{\operatorname{sign}(\theta_2)\,(\theta_1 - \theta_2)}{\pi}\,, \end{aligned} \tag{3.1}$$

where θ_1, $\theta_2 \in [-\pi, \pi]$ are the end tangent angles measured from the positive x-axis, see Figure 3.11. These constraints remove all of the bad curves in the middle of the (ϕ_1, ϕ_2)-plot, see Figure 3.12d. This region only contains good curves with $e_\lambda \leq 0.4$ which corresponds to an L^2 distance less than 0.01. We refer to this region for (ϕ_1, ϕ_2) as the *projection zone*. From this observation we now add the constraint that (ϕ_1, ϕ_2) should be contained in the projection zone. The boundary of the projection zone is given by interpolating the set of points $\{(1.05, 1.05), (1.9, 1.3), (\pi, 1.35), (4.3, 1.3), (5.2, 2\pi - 5.2)\}$ with a cubic spline interpolation and reflecting this curve around the lines $\phi_1 = \phi_2$ and $\phi_1 = 2\pi - \phi_2$.

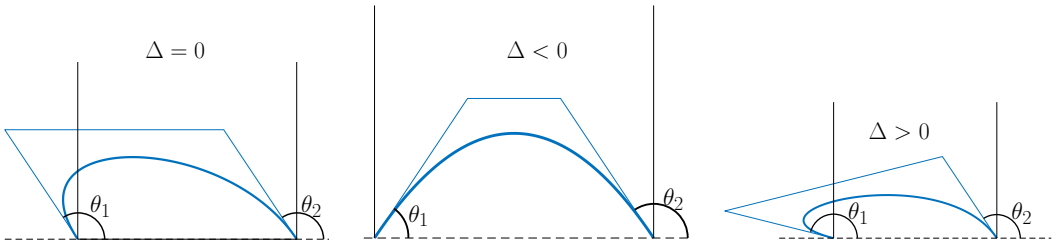


Fig. 3.11.: The length constraints and Δ.

In Table 3.3 and 3.4 we summarize the statistics for the projection zone. Adding this as a constraint to the length and angle constraints, we have a criterion for expected closeness to an elastic curve given by these constraints that all should be met for a curve with no self-intersecting control polygon:

- Angle Constraints 1–2
- Length Constraints 1–2
- The (ϕ_1, ϕ_2) Constraint. That is (ϕ_1, ϕ_2) is contained in the projection zone.

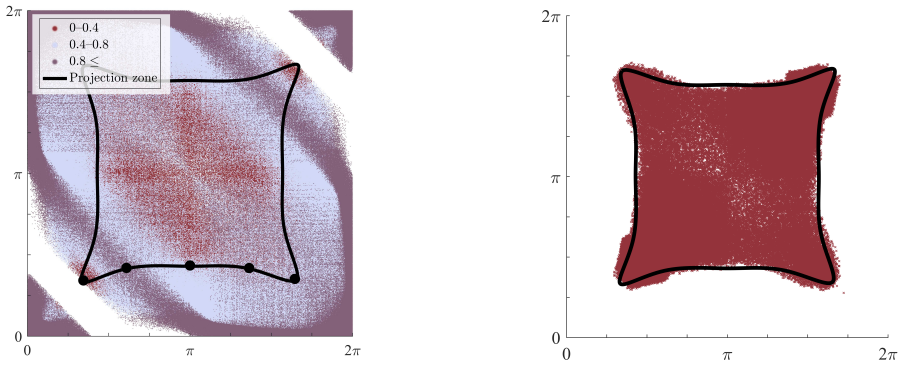
To verify that the above criterion only gives us curves with $e_\lambda \leq 0.4$ we have generated a sample of 10.7 million cubic Bézier curves, satisfying the constraints, and computed the λ-residuals. For this sample, we obtained a maximum of $e_\lambda = 0.393$. Because the λ-residual depends continuously on the control points and the projection zone is a compact set of $\mathbb{R}^4$, we can with a high likelihood conclude that the curves have $e_\lambda \leq 0.4$. To verify that the curves are visually close to the elastic curves we took another randomized sample of 38,826 curves satisfying the above criterion. For each of these curves, we computed the best approximating elastic curve using the normalized L^2 and H^1 distances, (3.2) and (3.3). In Figure 3.13 and 3.14 we show the results as two plots. From the plots, we conclude that we get L^2 and H^1 distances bounded by respectively 0.01 (our threshold) and 0.12.

Tab. 3.3.: Statistic for the sample satisfying the angle constraints in the projection zone.

# in proj. zone	# $e_\lambda \leq 0.4$	$e_\lambda \in (0.4, 0.8)$	$0.8 \leq e_\lambda$	% $e_\lambda < 0.8$	Mean	Max
961,731	666,287	261,802	33,642	96.5%	0.32	1

Tab. 3.4.: Statistic for the sample satisfying the angle and length constraints in the projection zone.

# in proj. zone	# $e_\lambda < 0.22$	$e_\lambda \in [0.22, 0.4]$	$0.4 < e_\lambda$	Mean	Max
188,043	155,389	32,654	0	0.15	0.4



(a) The color indicates e_λ. No length constraints imposed.

(b) Length constraints are satisfied and $e_\lambda \leq 0.22$.

(c) Length constraints are satisfied and $0.22 \leq e_\lambda \leq 0.4$.

(d) Length constraints are satisfied and $0.4 < e_\lambda$.

Fig. 3.12.: The (ϕ_1, ϕ_2)-plots for curves satisfying the angle constraints, with and without Length Constraints 1–2 . The black dots in 3.12a are used to define the projection zone.

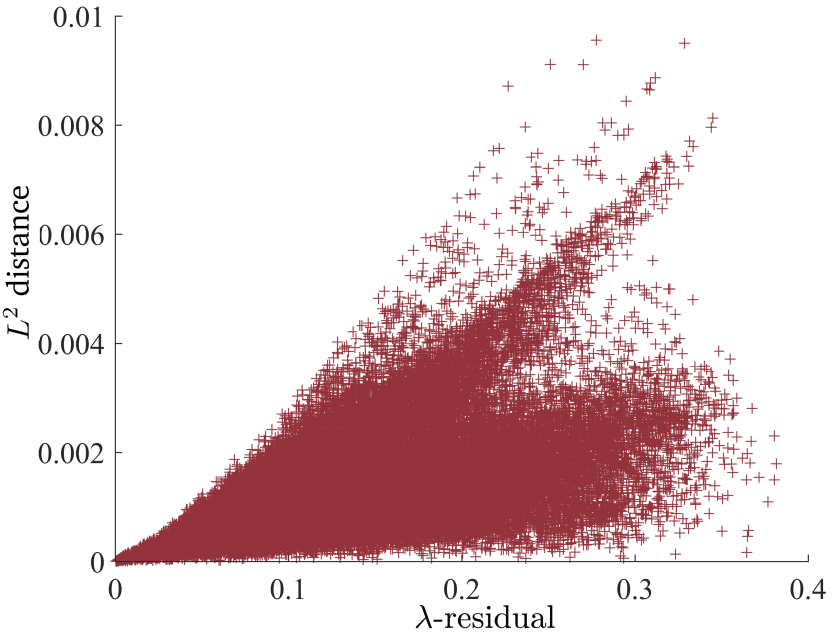


Fig. 3.13.: The curves that satisfy the criterion are close to the elastic curves in the normalized L^2 norm. The x-axis is the λ-residual e_λ (2.3).

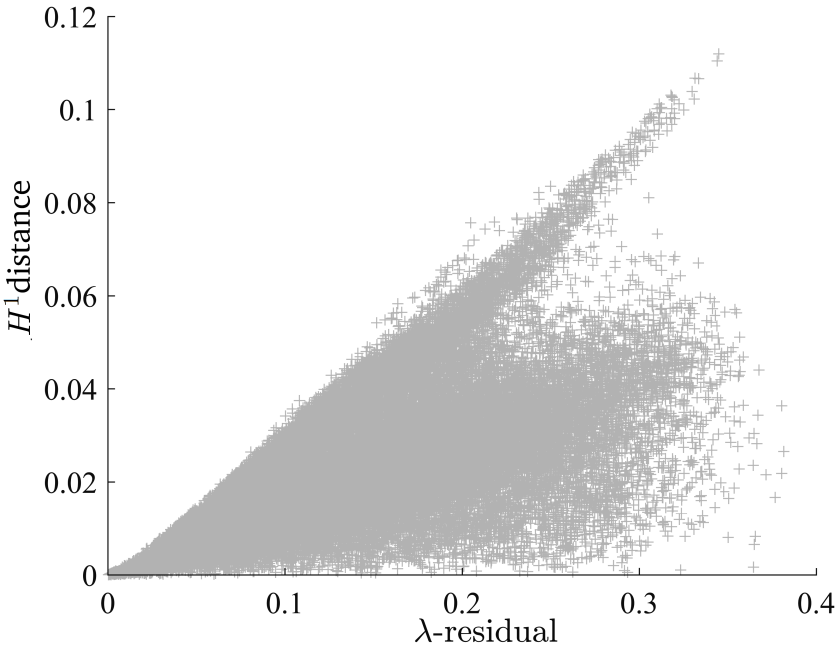


Fig. 3.14.: The curves that satisfy the criterion are close to the elastic curves in the H^1 norm. The x-axis is the λ-residual e_λ (2.3).

3.3 The connection between $\kappa = \lambda_2 x - \lambda_1 y + \alpha$ and the closeness to an elastic curve

In the above discussion, we have used the normalized L^2 distance as a measure for closeness to an elastica. We compute the normalized L^2 distance between the curve $\gamma : [0,1] \to \mathbb{R}^2$ and the approximating elastica γ_e as

$$\sqrt{\int_0^1 \frac{\|\gamma(t) - \gamma_e(s(t)/L)\|^2}{L^3} \|\gamma'(t)\| \, \mathrm{d}t}\,, \tag{3.2}$$

where L is the curve length and s is the arc length function of γ. Instead of the L^2 distance, a different measure is the H^1 distance given by

$$\sqrt{\int_0^1 \frac{\|\gamma(t) - \gamma_e(s(t)/L)\|^2}{L^3} \|\gamma'(t)\| \, \mathrm{d}t + \int_0^1 (\theta(t) - \theta_e(s(t)/L))^2 \frac{\|\gamma'(t)\|}{L} \, \mathrm{d}t}\,, \tag{3.3}$$

where θ and θ_e are the tangent angle functions of respectively γ and γ_e. If we compute the best elastic curve in the H^1 distance to each curve in the 80,000 sample, then we get a similar plot for the λ-residual e_λ (2.3), see Figure 3.15. Until now we have only used the L^2 and H^1 distances because they are easy to compute, whereas distances such as the Hausdorff distance are more time expensive.

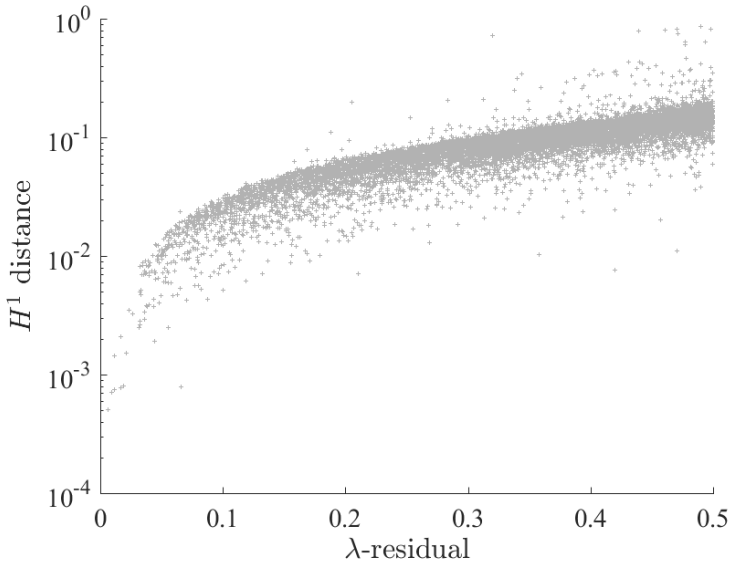


Fig. 3.15.: The H^1 distance and the λ-residual.

Motivated by the plots, Figure 3.7 and 3.15, we explain why a small λ-residual of a curve gives us little visual difference to an elastica. Our starting point is the ODE (ordinary differential equation) of an elastica $\gamma_e = (x, y) : [0, L] \to \mathbb{R}^2$ given by:

$$\kappa = \lambda_2 x - \lambda_1 y + \alpha\,,$$

where κ denotes the curvature and $\lambda_1, \lambda_2, \alpha \in \mathbb{R}$. This equation is obtained by applying the method of Lagrange multipliers to the boundary value problem. For more details we refer the reader to [Bra+17]. Without loss of generality, we assume that all curves in the following discussion (and next subsection) start at $(0,0)$ and have unit speed. With these assumptions, we write the ODE as

$$\theta_e'(s) = \lambda_2 \int_0^s \cos\theta_e \,\mathrm{d}t - \lambda_1 \int_0^s \sin\theta_e \,\mathrm{d}t + \alpha\,,$$

where θ_e is the tangent angle function of the elastica γ_e. Given λ_1, λ_2, and α, we consider how well an *arbitrary* arc-length parameterized $\gamma \in C^2([0,L],\mathbb{R}^2)$, $\gamma(0) = (0,0)^T$, satisfies this equation. We let $\theta \in C^1([0,L],\mathbb{R})$ denote the tangent angle of γ, set $\lambda = (\lambda_1, \lambda_2, \alpha)$, and define the residual of the ODE as $r : C^1([0,L],\mathbb{R}) \to C^0([0,L],\mathbb{R})$:

$$r(\theta)(s) = \theta'(s) - f_\lambda(\theta, s)\,,$$

where

$$f_\lambda(\theta, s) = \lambda_2 \int_0^s \cos\theta \,\mathrm{d}t - \lambda_1 \int_0^s \sin\theta \,\mathrm{d}t + \alpha\,.$$

This residual measures how close θ is to satisfying the ODE of the elastica γ_e. Having defined r, we determine inequalities between $\|r(\theta)\|_2$ and the distance between γ and γ_e in various norms. We will use these results to determine an inequality between the λ-residual e_λ and the normalized L^2 distance. The basic tool for the results is:

Theorem 1. *With the above notations, $\theta_e(0) = \theta(0)$, and $s_0 \in [0,L]$ such that $\|\theta - \theta_e\|_\infty = |(\theta - \theta_e)(s_0)|$. Then*

$$\|\theta - \theta_e\|_\infty \leq F(\eta)\sqrt{L}\|r(\theta)\|_2\,,$$

where $\eta = s_0^2\sqrt{\lambda_1^2 + \lambda_2^2}$ (scale-invariant) and the function F is defined as

$$F(\eta) = \begin{cases} \left(\left(\frac{2}{2-\eta}\right)^2 - 1\right)\frac{\sqrt{2}}{\eta}, & \eta \leq 1 \\ \left(2^{2\eta+1} - 1\right)\sqrt{\frac{2}{\eta}}, & \textit{otherwise} \end{cases}. \tag{3.4}$$

In Theorem 1 we use the standard notations $\|f\|_\infty = \sup_{s\in[0,L]}|f|$ and $\|f\|_2 = \sqrt{\int_0^L f^2 \,\mathrm{d}s}$. We explain the choice of F in the proof of Theorem 1. The essential property is that F is bounded for bounded η (the reader can check the case $\eta = 0$ with L'Hôpital's rule). By Theorem 1 and Corollary 1 below we then get, for η bounded, that we can bound $\|\theta - \theta_e\|_\infty$ and $\|\gamma - \gamma_e\|_\infty / L$ by $\|r\|_2$. Before we prove Theorem 1 we need Lemma 1 below. For $s_0 \in [0,L]$ we define the function $T_{s_0}(\theta) : C^1([0,L],\mathbb{R}) \to C^1([s_0,L],\mathbb{R})$ as

$$(T_{s_0}(\theta))(s) = \theta(s_0) + \int_{s_0}^s f_\lambda(\theta, t)\,\mathrm{d}t\,.$$

Lemma 1. *Given* $\theta_1, \theta_2 \in C^1([0,L],\mathbb{R})$ *choose* $s_0 \in [0,L]$ *such that* $\|\theta_1-\theta_2\|_\infty = |\theta_1(s_0)-\theta_2(s_0)|$. *Then for all* $0 \le \varepsilon \le 1$ *we have*

$$|(T_{s_0-s_0\varepsilon}(\theta_1) - T_{s_0-s_0\varepsilon}(\theta_2))(s_0)| \le |(\theta_1-\theta_2)(s_0-s_0\varepsilon)| + \varepsilon\eta\,|(\theta_1-\theta_2)(s_0)|\,,$$

where $\eta = s_0^2\sqrt{\lambda_1^2+\lambda_2^2}$.

Proof. As $|\lambda_1\cos c - \lambda_2 \sin c| \le \sqrt{\lambda_1^2+\lambda_2^2}$ for all $c\in\mathbb{R}$, we obtain by an application of the mean-value theorem to the function $g(\theta) = \lambda_2\cos\theta - \lambda_1\sin\theta$ for all $t\in[0,L]$:

$$\begin{aligned}
|f_\lambda(\theta_1,t) - f_\lambda(\theta_2,t)| &= \left|\int_0^t (\lambda_2\cos\theta_1 - \lambda_1\sin\theta_1) - (\lambda_2\cos\theta_2 - \lambda_1\sin\theta_2)\,\mathrm{d}s\right| \\
&\le \int_0^t |(\lambda_2\cos\theta_1 - \lambda_1\sin\theta_1) - (\lambda_2\cos\theta_2 - \lambda_1\sin\theta_2)|\,\mathrm{d}s \\
&\le \sqrt{\lambda_1^2+\lambda_2^2}\int_0^t |\theta_1-\theta_2|\,\mathrm{d}s\,.
\end{aligned}$$

The lemma then follows from the triangle inequality:

$$\begin{aligned}
|(T_{s_0-s_0\varepsilon}(\theta_1) - T_{s_0-s_0\varepsilon}(\theta_2))(s_0)| &\le |(\theta_1-\theta_2)(s_0-s_0\varepsilon)| \\
&\quad + \left|\int_{s_0-s_0\varepsilon}^{s_0} f_\lambda(\theta_1,t) - f_\lambda(\theta_2,t)\,\mathrm{d}t\right| \\
&\le |(\theta_1-\theta_2)(s_0-s_0\varepsilon)| \\
&\quad + \int_{s_0-s_0\varepsilon}^{s_0} |f_\lambda(\theta_1,t) - f_\lambda(\theta_2,t)|\,\mathrm{d}t \\
&\le |(\theta_1-\theta_2)(s_0-s_0\varepsilon)| \\
&\quad + \int_{s_0-s_0\varepsilon}^{s_0} \sqrt{\lambda_1^2+\lambda_2^2}\int_0^t |\theta_1-\theta_2|\,\mathrm{d}s\,\mathrm{d}t \\
&\le |(\theta_1-\theta_2)(s_0-s_0\varepsilon)| + \varepsilon s_0^2\sqrt{\lambda_1^2+\lambda_2^2}\,|(\theta_1-\theta_2)(s_0)| \\
&= |(\theta_1-\theta_2)(s_0-s_0\varepsilon)| + \varepsilon\eta\,|(\theta_1-\theta_2)(s_0)|\,.
\end{aligned}$$

□

With the lemma we are now ready to prove Theorem 1.

Proof (Theorem 1). For $0<\varepsilon<\frac{1}{\eta}$ and $\varepsilon\le 1$ we obtain

$$\begin{aligned}
|(\theta-\theta_e)(s_0)| &\le |(\theta - T_{s_0-s_0\varepsilon}(\theta))(s_0)| + |(T_{s_0-s_0\varepsilon}(\theta) - T_{s_0-s_0\varepsilon}(\theta_e))(s_0)| \\
&\quad + \underbrace{|(T_{s_0-s_0\varepsilon}(\theta_e) - \theta_e)(s_0)|}_{=0}
\end{aligned}$$

by Lemma 1

$$\le |(\theta - T_{s_0-s_0\varepsilon}(\theta))(s_0)| + |(\theta-\theta_e)(s_0-s_0\varepsilon)| + \varepsilon\eta|(\theta-\theta_e)(s_0)|$$

by insertion and definition of r

$$= \left| \int_{s_0 - s_0\varepsilon}^{s_0} r(\theta)\,\mathrm{d}t \right| + |(\theta-\theta_e)(s_0 - s_0\varepsilon)| + \varepsilon\eta |(\theta-\theta_e)(s_0)|$$

by Hölder's inequality

$$\leq \sqrt{\varepsilon s_0}\|r(\theta)\|_2 + |(\theta-\theta_e)(s_0 - s_0\varepsilon)| + \varepsilon\eta |(\theta-\theta_e)(s_0)|\,.$$

After isolating $|(\theta-\theta_e)(s_0)|$ we get

$$|(\theta-\theta_e)(s_0)| \leq \frac{\sqrt{\varepsilon s_0}\|r(\theta)\|_2 + |(\theta-\theta_e)(s_0 - s_0\varepsilon)|}{1-\varepsilon\eta}\,. \tag{3.5}$$

We generalize (3.5) by choosing $s_j, j \in \mathbb{N}$ such that $\left\|(\theta-\theta_e)|_{[0,s_0-js_0\varepsilon]}\right\|_\infty = |(\theta-\theta_e)(s_j)|$. For $s_j > s_0\varepsilon$ we get with the above chain of arguments (where $s_0\varepsilon/s_j$ is our new choice of ε)

$$|(\theta-\theta_e)(s_j)| \leq \frac{\sqrt{\varepsilon s_0}\|r(\theta)\|_2 + |(\theta-\theta_e)(s_j - s_0\varepsilon)|}{1 - \frac{s_0\varepsilon}{s_j} s_j^2\sqrt{\lambda_1^2+\lambda_2^2}} \leq \frac{\sqrt{\varepsilon s_0}\|r(\theta)\|_2 + |(\theta-\theta_e)(s_{j+1})|}{1-\varepsilon\eta}\,.$$

For $s_j \leq s_0\varepsilon$ we choose 1 as our new ε and get

$$|(\theta-\theta_e)(s_j)| \leq \frac{\sqrt{s_j}\|r(\theta)\|_2}{1 - s_j^2\sqrt{\lambda_1^2+\lambda_2^2}} \leq \frac{\sqrt{\varepsilon s_0}\|r(\theta)\|_2}{1-\varepsilon\eta}\,.$$

All in all we have

$$\left\|(\theta-\theta_e)|_{[0,s_0-js_0\varepsilon]}\right\|_\infty = |(\theta-\theta_e)(s_j)| \leq \begin{cases} \frac{\sqrt{\varepsilon s_0}\|r(\theta)\|_2 + |(\theta-\theta_e)(s_{j+1})|}{1-\varepsilon\eta}, & s_j > s_0\varepsilon \\ \frac{\sqrt{\varepsilon s_0}\|r(\theta)\|_2}{1-\varepsilon\eta}, & s_j \leq s_0\varepsilon \end{cases}.$$

If $s_0 \leq (j+1)s_0\varepsilon$ then $s_j \leq s_0\varepsilon$ and we get $|(\theta-\theta_e)(s_j)| \leq \frac{\sqrt{\varepsilon s_0}\|r(\theta)\|_2}{1-\varepsilon\eta}$. In the other case, $s_0 > (j+1)s_0\varepsilon$, then either $s_j \leq s_0\varepsilon$ or $s_j > s_0\varepsilon$, but in both cases $|(\theta-\theta_e)(s_j)| \leq \frac{\sqrt{\varepsilon s_0}\|r(\theta)\|_2 + |(\theta-\theta_e)(s_{j+1})|}{1-\varepsilon\eta}$, and hence we have proved:

$$\left\|(\theta-\theta_e)|_{[0,s_0-js_0\varepsilon]}\right\|_\infty = |(\theta-\theta_e)(s_j)| \leq \begin{cases} \frac{\sqrt{\varepsilon s_0}\|r(\theta)\|_2 + |(\theta-\theta_e)(s_{j+1})|}{1-\varepsilon\eta}, & s_0 > (j+1)s_0\varepsilon \\ \frac{\sqrt{\varepsilon s_0}\|r(\theta)\|_2}{1-\varepsilon\eta}, & s_0 \leq (j+1)s_0\varepsilon \end{cases}. \tag{3.6}$$

Using this we now let $j \in \mathbb{N}$ such that $j\varepsilon < 1 \leq (j+1)\varepsilon$, i.e., we let $j = \lceil 1/\varepsilon \rceil - 1$. From (3.6) we obtain by recursion a geometric sum with $\lceil 1/\varepsilon \rceil = j+1$ terms. That is

$$\|\theta-\theta_e\|_\infty = |(\theta-\theta_e)(s_0)| \leq \sum_{i=1}^{\lceil 1/\varepsilon \rceil} \left(\frac{1}{1-\varepsilon\eta}\right)^i \sqrt{\varepsilon s_0}\|r(\theta)\|_2\,.$$

To evaluate this sum we must choose ε such that $0 < \varepsilon < \frac{1}{\eta}$ and $\varepsilon \leq 1$. If $\eta \leq 1 \Leftrightarrow 1 \leq \frac{1}{\eta}$, we put $\varepsilon = 1/2$ (but other values work as well). With this choice we evaluate the sum

$$\begin{aligned}
\|\theta - \theta_e\|_\infty &\leq \sum_{i=1}^{2} \left(\frac{1}{1-\frac{\eta}{2}}\right)^i \sqrt{\frac{s_0}{2}}\|r(\theta)\|_2 \\
&\leq \frac{\frac{2}{2-\eta} - \left(\frac{2}{2-\eta}\right)^3}{1-\frac{2}{2-\eta}} \sqrt{\frac{L}{2}}\|r(\theta)\|_2 \\
&= \frac{2\left(\frac{2}{2-\eta}\right)^2 - 2}{\eta} \frac{1}{\sqrt{2}} \sqrt{L}\|r(\theta)\|_2 \\
&= F(\eta)\sqrt{L}\|r(\theta)\|_2 \,.
\end{aligned}$$

Otherwise, if $\eta > 1$, we put $\varepsilon = \frac{1}{2\eta}$ and obtain

$$\begin{aligned}
\|\theta - \theta_e\|_\infty \leq \sum_{i=1}^{\lceil 2\eta \rceil} \left(\frac{1}{1-\frac{1}{2}}\right)^i \sqrt{\frac{s_0}{2\eta}}\|r(\theta)\|_2 &\leq \sum_{i=1}^{\lceil 2\eta \rceil} 2^i \sqrt{\frac{L}{2\eta}}\|r(\theta)\|_2 \\
&\leq \left(2^{2\eta+2} - 2\right)\sqrt{\frac{L}{2\eta}}\|r(\theta)\|_2 \\
&= F(\eta)\sqrt{L}\|r(\theta)\|_2 \,.
\end{aligned}$$

□

In the above proof, we chose arbitrary values for ε to reach the expression for F. The task of determining the best ε, giving the strongest inequality, is non-trivial. Hence simplified deviations of the results in the proof would be convenient and might lead to a stronger inequality in Theorem 1. From Theorem 1 we obtain the following corollary that gives us an inequality between $\|r(\theta)\|_2$ and $\frac{\|\gamma - \gamma_e\|_\infty}{L}$.

Corollary 1. *With the above notations,* $\theta_e(0) = \theta(0)$*, and* $s_0 \in [0, L]$ *such that* $\|\theta - \theta_e\|_\infty = |(\theta - \theta_e)(s_0)|$*. Then*

$$\frac{\|\gamma - \gamma_e\|_\infty}{L} \leq F(\eta)\sqrt{L}\|r(\theta)\|_2 \,,$$

where $\eta = s_0^2\sqrt{\lambda_1^2 + \lambda_2^2}$ *and the function* $F(\eta)$ *is given by* (3.4)*.*

Proof. Using $(\cos\theta - \cos\theta_e)^2 + (\sin\theta - \sin\theta_e)^2 \leq (\theta - \theta_e)^2$ we obtain for some $s \in [0, L]$:

$$\begin{aligned}
\|\gamma - \gamma_e\|_\infty &= \|\gamma(s) - \gamma_e(s)\| \\
&= \sqrt{\left(\int_0^s \cos\theta - \cos\theta_e \,\mathrm{d}t\right)^2 + \left(\int_0^s \sin\theta - \sin\theta_e \,\mathrm{d}t\right)^2}
\end{aligned}$$

$$
\begin{aligned}
&\leq \sqrt{s\int_0^s (\cos\theta-\cos\theta_e)^2\,\mathrm{d}t + s\int_0^s (\sin\theta-\sin\theta_e)^2\,\mathrm{d}t}\\
&\leq \sqrt{L\int_0^L (\theta-\theta_e)^2\,\mathrm{d}t}\\
&\leq L\|\theta-\theta_e\|_\infty\,,
\end{aligned}
$$

and the inequality follows from $\|\theta-\theta_e\|_\infty \leq F(\eta)\sqrt{L}\|r(\theta)\|_2$. □

With Theorem 1 and Corollary 1 we reach an understanding of the previous plot for e_λ and the L^2 distance. If λ_1, λ_2, and α are the constants determined by the initial guess algorithm, then $e_\lambda = \|r(\theta)\|_2 \Big/ \sqrt{\int \kappa^2\,\mathrm{d}s}$ and from Corollary 1 we obtain

$$
\frac{\|\gamma-\gamma_e\|_2}{L^{3/2}} \leq F(\eta)\sqrt{L}\sqrt{\int_0^L \kappa^2\,\mathrm{d}s}\,e_\lambda\,. \tag{3.7}
$$

3.3.1 Residuals for measuring the closeness to an elastic curve

Instead of e_λ we can get other scale-invariant residuals for measuring the closeness to an elastica. If we use $e_r = \sqrt{L}\|r(\theta)\|_2$, then we obtain a simpler inequality for the L^2 distance

$$
\frac{\|\gamma-\gamma_e\|_2}{L^{3/2}} \leq F(\eta)e_r\,.
$$

From this, inequality (3.7), Theorem 1, and Corollary 1, we get inequalities between the distance to an elastica and the residual of the ODE. With the inequalities, we can, given a set of curves, advantageously restrict e_λ or e_r to obtain a subset visually close to the elastica, provided that

- the scale invariant factors F and $F\sqrt{L}\sqrt{\int \kappa^2\,\mathrm{d}s}$ are bounded. That is there exists a threshold for e_r or e_λ that guarantees closeness to an elastica and
- with this threshold we have a sufficiently large set of curves.

From the definition of F and η, we get that we bound F by bounding $\ell = L^2\sqrt{\lambda_1^2+\lambda_2^2}$. For elastic curves the value $\sqrt{\ell}$ is the length of the basic elastic curve segment ξ_k. To get a geometrical understanding of the curves with bounded ℓ, we prove Theorem 2 below.

Theorem 2. *With the above notations, assuming λ_1, λ_2 are computed with the initial guess algorithm for $\gamma = (x, y)$, we have*

$$
\ell = \left\| M_{2\times 3}^{-1}\left(-\int_0^1 \bar{y}\bar{\kappa}\,\mathrm{d}s, \int_0^1 \bar{x}\bar{\kappa}\,\mathrm{d}s, \int_0^1 \bar{\kappa}\,\mathrm{d}s\right)^T \right\|,
$$

where $(\bar{x}, \bar{y})$ and $\bar{\kappa}$ denote respectively the coordinates and curvature of γ normalized to have curve length 1. $M^{-1}_{2\times 3}$ is the 2×3 matrix given by the first two rows of M^{-1} where

$$M = \begin{pmatrix} \int_0^1 \bar{y}^2 \,\mathrm{d}s & -\int_0^1 \bar{x}\bar{y} \,\mathrm{d}s & -\int_0^1 \bar{y} \,\mathrm{d}s \\ -\int_0^1 \bar{x}\bar{y} \,\mathrm{d}s & \int_0^1 \bar{x}^2 \,\mathrm{d}s & \int_0^1 \bar{x} \,\mathrm{d}s \\ -\int_0^1 \bar{y} \,\mathrm{d}s & \int_0^1 \bar{x} \,\mathrm{d}s & 1 \end{pmatrix}.$$

Proof. The values λ_1, λ_2 and α are computed as the solution to the linear system

$$\underbrace{\begin{pmatrix} \int_0^L y^2 \,\mathrm{d}s & -\int_0^L xy \,\mathrm{d}s & -\int_0^L y \,\mathrm{d}s \\ -\int_0^L xy \,\mathrm{d}s & \int_0^L x^2 \,\mathrm{d}s & \int_0^L x \,\mathrm{d}s \\ -\int_0^L y \,\mathrm{d}s & \int_0^L x \,\mathrm{d}s & L \end{pmatrix}}_{=M_L} \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \alpha \end{pmatrix} = \begin{pmatrix} -\int_0^L y\kappa \,\mathrm{d}s \\ \int_0^L x\kappa \,\mathrm{d}s \\ \int_0^L \kappa \,\mathrm{d}s \end{pmatrix}.$$

The matrix M_L satisfies

$$M_L = M \circ \begin{pmatrix} L^3 & L^3 & L^2 \\ L^3 & L^3 & L^2 \\ L^2 & L^2 & L \end{pmatrix},$$

where $\circ$ is the entrywise product. With the usual formula for the inverse of a 3×3 matrix we have

$$M_L^{-1} = M^{-1} \circ \begin{pmatrix} L^{-3} & L^{-3} & L^{-2} \\ L^{-3} & L^{-3} & L^{-2} \\ L^{-2} & L^{-2} & L^{-1} \end{pmatrix}.$$

Now the first two rows of M_L^{-1} is given by

$$M^{-1}_{L,2\times 3} = M^{-1}_{2\times 3} \circ \begin{pmatrix} L^{-3} & L^{-3} & L^{-2} \\ L^{-3} & L^{-3} & L^{-2} \end{pmatrix},$$

and by change of variables in the integrations we get

$$(\lambda_1, \lambda_2)^T = M^{-1}_{L,2\times 3} \left(-\int_0^L y\kappa \,\mathrm{d}s, \int_0^L x\kappa \,\mathrm{d}s, \int_0^L \kappa \,\mathrm{d}s \right)^T$$

$$
\begin{aligned}
&= \left(M_{2\times 3}^{-1} \circ \begin{pmatrix} L^{-3} & L^{-3} & L^{-2} \\ L^{-3} & L^{-3} & L^{-2} \end{pmatrix} \right) \left(-\int_0^L y\kappa \,\mathrm{d}s, \int_0^L x\kappa \,\mathrm{d}s, \int_0^L \kappa \,\mathrm{d}s \right)^T \\
&= \left(M_{2\times 3}^{-1} \circ \begin{pmatrix} L^{-3} & L^{-3} & L^{-2} \\ L^{-3} & L^{-3} & L^{-2} \end{pmatrix} \right) \left(-L\int_0^1 \bar{y}\bar{\kappa} \,\mathrm{d}s, L\int_0^1 \bar{x}\bar{\kappa} \,\mathrm{d}s, \int_0^1 \bar{\kappa} \,\mathrm{d}s \right)^T \\
&= \frac{1}{L^2} M_{2\times 3}^{-1} \left(-\int_0^1 \bar{y}\bar{\kappa} \,\mathrm{d}s, \int_0^1 \bar{x}\bar{\kappa} \,\mathrm{d}s, \int_0^1 \bar{\kappa} \,\mathrm{d}s \right)^T .
\end{aligned}
$$

□

Because the entries of M are bounded, we obtain that ℓ is bounded for a set of curves if $\|\bar{k}\|_\infty = L\|k\|_\infty < \upsilon < \infty$ and $|\det(M)| > \tau > 0$. The value $|\det(M)|$ is 0 if and only if the curve is a line segment. Let $D : C([0,1],\mathbb{R}^2) \to \mathbb{R}$ be the absolute value of the determinant given by the map

$$
\gamma = (x,y) \mapsto \left| \det \left(\begin{pmatrix} \int_0^1 y^2 \,\mathrm{d}t & -\int_0^1 xy \,\mathrm{d}t & -\int_0^1 y \,\mathrm{d}t \\ -\int_0^1 xy \,\mathrm{d}t & \int_0^1 x^2 \,\mathrm{d}t & \int_0^1 x \,\mathrm{d}t \\ -\int_0^1 y \,\mathrm{d}t & \int_0^1 x \,\mathrm{d}t & 1 \end{pmatrix} \right) \right| ,
$$

then we have:

Lemma 2. *$D(\gamma) = 0$ if and only if $\gamma = (x,y)$ is a line segment in $C([0,1],\mathbb{R}^2)$.*

Proof. The implication "$\Leftarrow$" follows by insertion. To prove "$\Rightarrow$" we assume that the determinant is 0. In this case we have a non-trivial minimizer $(\beta_1, \beta_2, \delta)$ to

$$
\min_{\beta_1,\beta_2,\delta} \int_0^1 (\beta_1 x - \beta_2 y + \delta)^2 \,\mathrm{d}t \,.
$$

As the minimizer $(\beta_1, \beta_2, \delta)$ to this problem satisfies $(\beta_1 x - \beta_2 y + \delta)^2 = 0$, we obtain $\beta_1 x - \beta_2 y + \delta = 0$. □

Using this lemma we prove Theorem 3 below. In Theorem 3 we use the notations:

$$
\begin{aligned}
U &= \{\gamma \in C^2([0,1],\mathbb{R}^2) \,|\, \|\gamma'(t)\| = 1 \text{ for all } t \text{ and } \gamma(0) = (0,0)^T\} \,, \\
B(l,\varepsilon) &= \{\gamma \in C([0,1],\mathbb{R}^2) \,|\, \|\gamma - l\|_\infty < \varepsilon\} \,,
\end{aligned}
$$

where l is a line segment in $C([0,1],\mathbb{R}^2)$ (we denote the set of line segments as $\mathcal{L}$).

Theorem 3. *For $\varepsilon > 0$ there exists $\tau > 0$ such that for all $\gamma \in \mathcal{F}_\varepsilon = U \setminus \bigcup_{l\in\mathcal{L}} B(l,\varepsilon)$*

$$
D(\gamma) > \tau > 0 \,.
$$

To prove Theorem 3, we first show that the closure of $\mathcal{F}_\varepsilon$ in $C([0,1],\mathbb{R}^2)$, equipped with the sup metric, is compact. With this result, we prove Theorem 3 by an

application of the extreme value theorem, where we use $D(\gamma) = 0 \Leftrightarrow \gamma \in \mathcal{L}$ and the continuity of D.

Lemma 3. *The subspace $\mathcal{F}_\varepsilon$ has compact closure in $C([0,1],\mathbb{R}^2)$ with the metric induced by the norm $\|f\|_\infty = \sup\|f\|$.*

Proof. We apply Ascoli's theorem [Mun00] that gives us the result if and only if:

- $\mathcal{F}_\varepsilon$ is equicontinuous under $\|-\|$ and
- $\mathcal{F}_\varepsilon$ is pointwise bounded under $\|-\|$.

Recall that the set $\mathcal{F}_\varepsilon$ is equicontinuous at $t_0 \in [0,1]$ under $\|-\|$ if given $\epsilon > 0$ there exists $\delta > 0$ such that for all $t \in (t_0 - \delta, t_0 + \delta)$ and all $\gamma \in \mathcal{F}_\varepsilon$,

$$\|\gamma(t) - \gamma(t_0)\| < \epsilon\,.$$

To prove equicontinuity we therefore let $t_0 \in [0,1]$ and $\epsilon > 0$. If we choose $\delta = \epsilon/2$, then we get by the mean value theorem for all $t \in (t_0 - \delta, t_0 + \delta)$ and $\gamma \in \mathcal{F}_\varepsilon$,

$$\|\gamma(t) - \gamma(t_0)\| \leq |t - t_0| \leq \frac{\epsilon}{2} < \epsilon\,,$$

and we thereby obtain equicontinuity of $\mathcal{F}$.

By definition $\mathcal{F}_\varepsilon$ is pointwise bounded if for each $t \in [0,1]$,

$$\sup\{\|\gamma(t)\| \,|\, \gamma \in \mathcal{F}_\varepsilon\} < \infty\,.$$

This holds for $\mathcal{F}_\varepsilon$ because $\gamma \in \mathcal{F}_\varepsilon$ has length 1 and $\gamma(0) = (0,0)^T$, so $\|\gamma(t)\| \leq 1$ for all $\gamma \in \mathcal{F}_\varepsilon$ and $t \in [0,1]$. Hence we have verified that $\mathcal{F}_\varepsilon$ is equicontinuous and pointwise bounded under $\|-\|$. By Ascoli's theorem $\mathcal{F}_\varepsilon$ has compact closure in $C([0,1],\mathbb{R}^2)$. □

To prove Theorem 3 we need to show that $D : C([0,1],\mathbb{R}^2) \to \mathbb{R}$ is continuous.

Lemma 4. *$D : C([0,1],\mathbb{R}^2) \to \mathbb{R}$ is continuous.*

Proof. Because the determinant is a polynomial in the values $\int x\,\mathrm{d}t$, $\int y\,\mathrm{d}t$, $\int xy\,\mathrm{d}t$, $\int x^2\,\mathrm{d}t$, and $\int y^2\,\mathrm{d}t$, it is sufficient to show that

$$\gamma = (x,y) \mapsto \int_0^1 xy\,\mathrm{d}t$$

is continuous. We prove this by an ϵ-δ argument. For $\gamma_1 = (x_1, y_1) \in C([0,1], \mathbb{R}^2)$ and any $\epsilon > 0$, we prove there exists $\delta > 0$ such that for all $\gamma = (x, y) \in C([0,1], \mathbb{R}^2)$ where $\|\gamma_1 - \gamma\|_\infty < \delta$,

$$\left| \int_0^1 x_1 y_1 \, \mathrm{d}t - \int_0^1 xy \, \mathrm{d}t \right| < \epsilon \, .$$

We first observe that $\|\gamma_1 - \gamma\|_\infty < \delta$ implies $\|x_1 - x\|_\infty, \|y_1 - y\|_\infty < \delta$. From this we obtain

$$\begin{aligned}
\left| \int_0^1 x_1 y_1 \, \mathrm{d}t - \int_0^1 xy \, \mathrm{d}t \right| &\leq \|x_1 y_1 - x_1 y + x_1 y - xy\|_\infty \\
&\leq \|x_1\|_\infty \|y_1 - y\|_\infty + \|y\|_\infty \|x_1 - x\|_\infty \\
&= \|x_1\|_\infty \|y_1 - y\|_\infty + \|y_1 + (y - y_1)\|_\infty \|x_1 - x\|_\infty \\
&\leq \|x_1\|_\infty \|y_1 - y\|_\infty + \|y_1\|_\infty \|x_1 - x\|_\infty + \|y_1 - y\|_\infty \|x_1 - x\|_\infty \\
&< \|x_1\|_\infty \delta + \|y_1\|_\infty \delta + \delta^2 \\
&= \left(\|x_1\|_\infty + \|y_1\|_\infty \right) \delta + \delta^2 \, ,
\end{aligned}$$

and if we choose

$$\delta < \frac{\sqrt{\left(\|x_1\|_\infty + \|y_1\|_\infty \right)^2 + 4\epsilon} - \left(\|x_1\|_\infty + \|y_1\|_\infty \right)}{2} \, ,$$

then we get $\left| \int_0^1 x_1 y_1 \, \mathrm{d}t - \int_0^1 xy \, \mathrm{d}t \right| < \epsilon$, and hence we have proved continuity of D. □

With the above lemmas, we are now able to prove Theorem 3.

Proof (Theorem 3). Because D is a continuous on $\overline{\mathcal{F}_\varepsilon}$ we obtain from the extreme value theorem that there exists $f \in \overline{\mathcal{F}_\varepsilon}$ such that for all $\gamma \in \overline{\mathcal{F}_\varepsilon}$:

$$D(f) \leq D(\gamma) \, .$$

Furthermore, we have $0 < D(f)$ because $\overline{\mathcal{F}_\varepsilon}$ does not contain line segments:

$$\overline{\mathcal{F}_\varepsilon} \subseteq \overline{U} \cap \overline{C([0,1], \mathbb{R}^2) \setminus \bigcup_{l \in \mathcal{L}} B(l, \varepsilon)} = \overline{U} \cap C([0,1], \mathbb{R}^2) \setminus \bigcup_{l \in \mathcal{L}} B(l, \varepsilon) \, .$$

By letting $\tau = D(f)/2$ the theorem follows. □

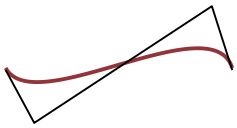
As Theorem 2 gives us $\ell = \left\| M_{2\times 3}^{-1} \left(-\int_0^1 \bar{y}\bar{\kappa} \, \mathrm{d}s, \int_0^1 \bar{x}\bar{\kappa} \, \mathrm{d}s, \int_0^1 \bar{\kappa} \, \mathrm{d}s \right)^T \right\|$, we can bound ℓ for a set of curves if we

- impose an upper bound on the scale invariant curvature $\|\bar{k}\|_\infty = L\|k\|_\infty$ and

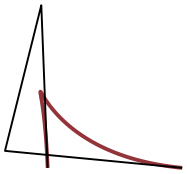
- impose a lower bound on $|\det(M)|$. We impose a lower bound on $|\det(M)|$ by discarding the curves that are, normalized with the curve length, close to a line segment in the uniform norm (Theorem 3).

In Figure 3.16 we include four examples of curves from the 80,000 sample with large ℓ.

(a) Curve with $\ell = 93.77$, $e_\lambda = 0.68$, $\det(M) = 2.12 \cdot 10^{-5}$ and $L\|\kappa\|_\infty = 12.52$.



(b) Curve with $\ell = 80.53$, $e_\lambda = 0.20$, $\det(M) = 4.06 \cdot 10^{-5}$ and $L\|\kappa\|_\infty = 10.56$.



(c) Curve with $\ell = 83.56$, $e_\lambda = 1.00$, $\det(M) = 1.23 \cdot 10^{-4}$ and $L\|\kappa\|_\infty = 4320.49$.

(d) Curve with $\ell = 271.62$, $e_\lambda = 0.99$, $\det(M) = 1.23 \cdot 10^{-5}$ and $L\|\kappa\|_\infty = 649.13$.

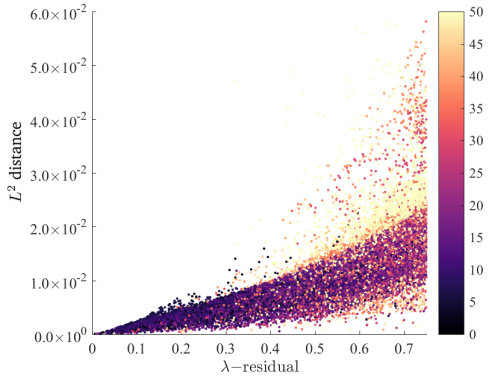
Fig. 3.16.: Examples of curves with large $\ell > 80$ from the 80,000 sample ($\ell = 80$ is the 80th percentile).

To see the effect of a bounded ℓ, we impose $\ell \leq 25$ on the 80,000 sample of cubic Bézier curves. This bound seems to give "cleaner" L^2 plots/less outliers, see Figure 3.17 and 3.18 for comparison. In Figure 3.19 we include some of the remaining outliers. In Figure 3.20 we plot e_λ and e_r against $\|\gamma - \gamma_e\|_\infty / L$ for curves satisfying $\ell \leq 25$. Here we have almost no outliers and still a large set of curves (we recover 77% of curves with normalized L^2 distance less than 0.01).

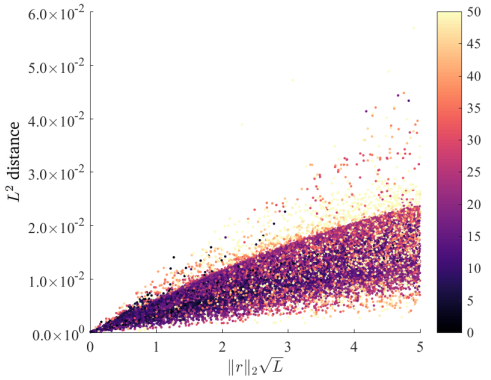
To obtain a more simple condition to $\ell \leq 25$ we have in MATLAB experimented with different inequalities, using $\|\kappa\|_\infty$ and L, such that $\ell \leq 25$ holds for the 80,000 sample. The following inequality seems to give us this:

$$\|\kappa\|_\infty L < 16\,(L-1)\,. \tag{3.8}$$

In Figure 3.21 we include, with this constraint, the plots for $\|\gamma - \gamma_e\|_\infty / L$ against e_λ and e_r. This constraint leaves us with 55% of the curves with L^2 distance less than 0.001.

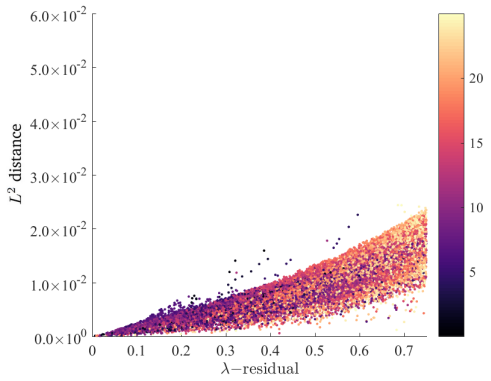


(a) Plot with respect to the λ-residual $e_\lambda = \|r(\theta)\|_2 \Big/ \sqrt{\int \kappa^2 \, \mathrm{d}s}$.

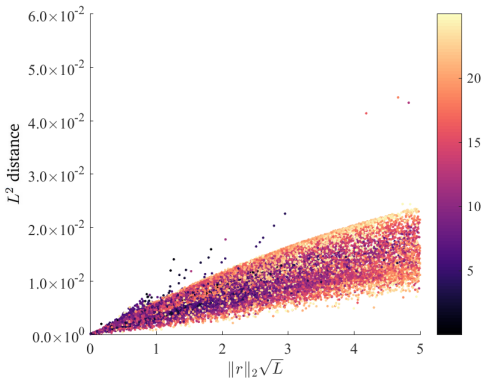


(b) Plot with respect to the residual $e_r = \|r(\theta)\|_2\sqrt{L}$.

Fig. 3.17.: The normalized L^2 distance. The color denotes ℓ.



(a) Plot with respect to the λ-residual and $\ell \leq 25$.



(b) Plot with respect to the residual $e_r = \|r(\theta)\|_2\sqrt{L}$ and $\ell \leq 25$.

Fig. 3.18.: The normalized L^2 distance with $\ell \leq 25$. The color denotes ℓ.

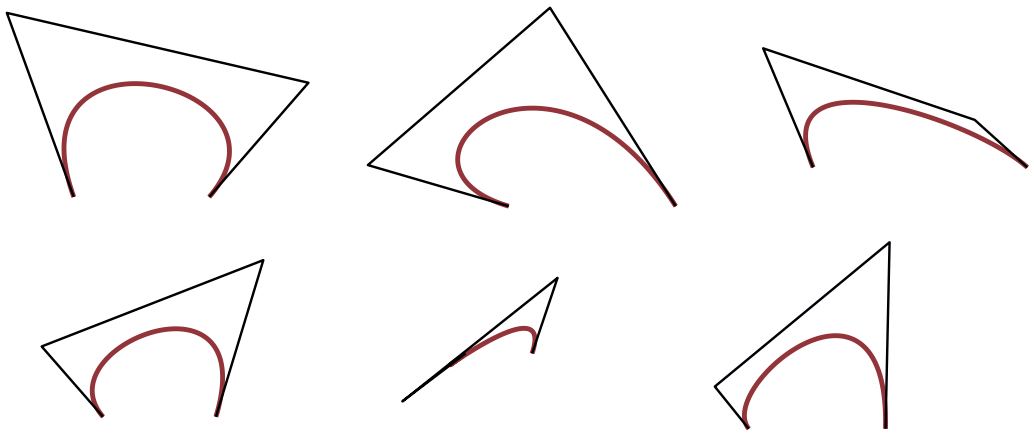
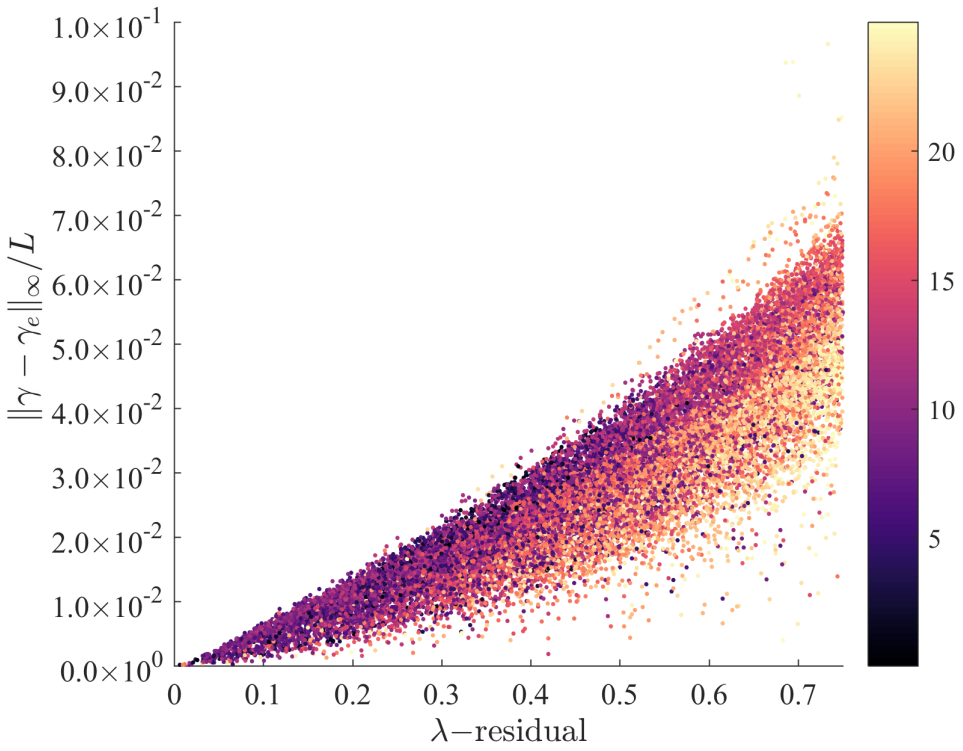
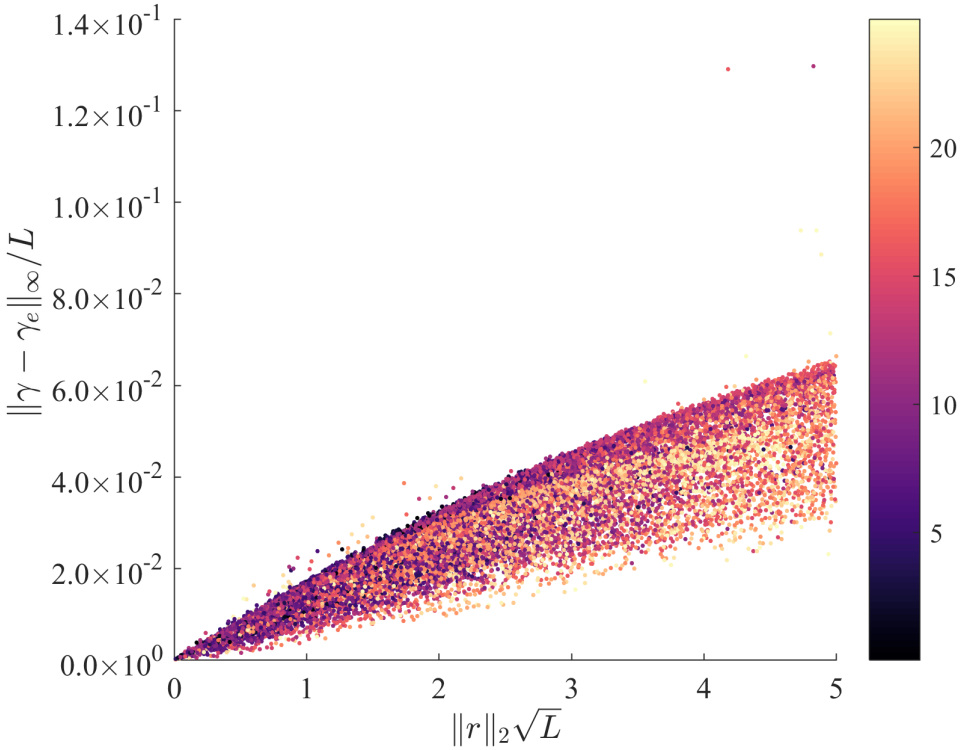


Fig. 3.19.: Outliers from Figure 3.18. The outliers (top row) are from the λ-residual vs. L^2 distance plot and the outliers on the bottom are from the e_r vs. L^2 distance plot.

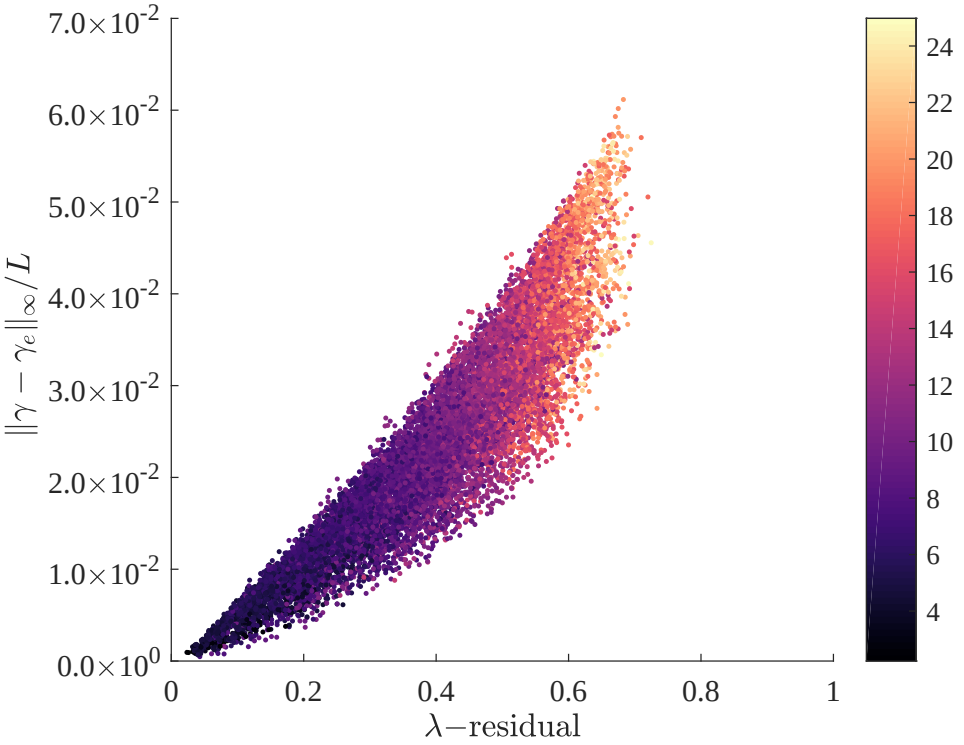


(a) Plot with respect to the λ-residual $e_\lambda = \|r(\theta)\|_2 \Big/ \sqrt{\int \kappa^2 \, \mathrm{d}s}$ and $\ell \leq 25$.

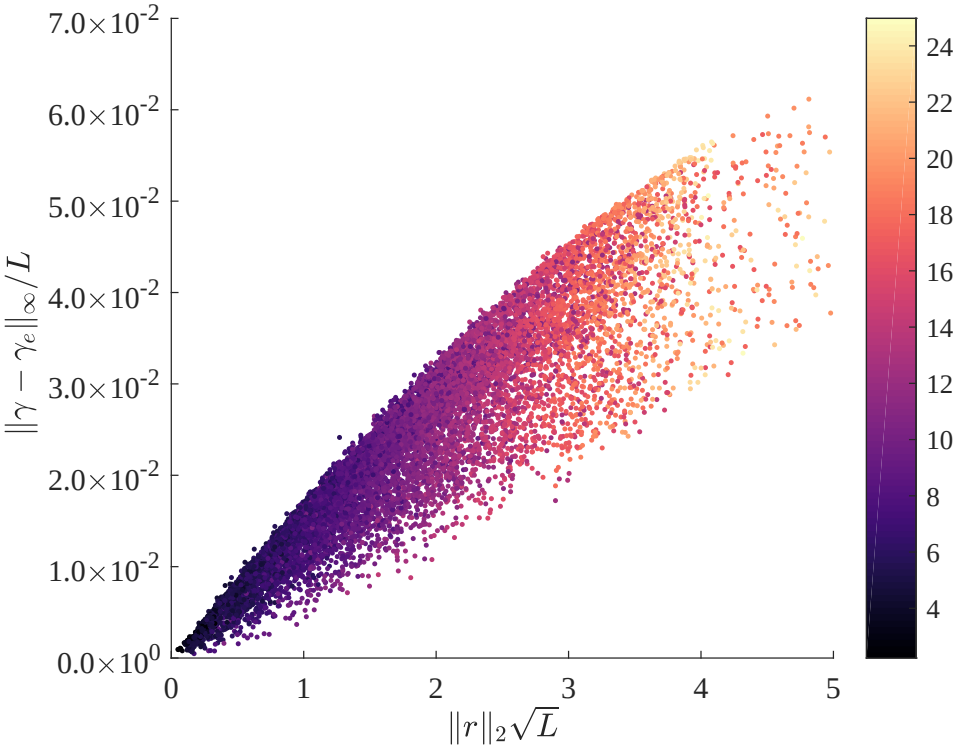


(b) Plot with respect to $e_r = \|r(\theta)\|_2\sqrt{L}$ and $\ell \leq 25$.

Fig. 3.20.: The $\|\gamma-\gamma_e\|_\infty/L$ values for curves satisfying $\ell \leq 25$. The color denotes ℓ.



(a) Plot with respect to the λ-residual $e_\lambda = \|r(\theta)\|_2 \Big/ \sqrt{\int \kappa^2 \, \mathrm{d}s}$



(b) Plot with respect to $e_r = \|r(\theta)\|_2 \sqrt{L}$.

Fig. 3.21.: The $\|\gamma - \gamma_e\|_\infty / L$ value for curves where $\|\kappa\|_\infty L < 16(L-1)$. The color denotes $\|\kappa\|_\infty L$.

The constraint (3.8) tells us that if L goes to 1 (the curve becomes close to a line[2]) then the scale invariant curvature must go to 0.

Summary

From the analysis of this section, we obtain that we can add bounds on $e_\lambda = \|r(\theta)\|_2 \Big/ \sqrt{\int \kappa^2 \, \mathrm{d}s}$ or $e_r = \sqrt{L}\|r(\theta)\|_2$ to restrict a set of curves to be visually close to elastica[3], provided that ℓ is bounded for the curves. The value ℓ is bounded for a set of curves that have bounded scale invariant curvature and are not close to a line segment in the uniform norm (normalized with the curve length), see Theorem 2 and 3.

From the experimental work, we also observe that, for our randomized sample of cubic Bézier curves, we can add a bound on e_λ or e_r to get visually close to an elastica (small L^2 or H^1 distance) and still have a large set of curves left. If we additionally impose $\ell \leq 25$, then we get cleaner plots between e_λ or e_r and the L^2 distance, see Figure 3.18. For $\|\gamma - \gamma_e\|_\infty / L$ we get that the constraint $\ell \leq 25$ or $\|\kappa\|_\infty L < 16\,(L-1)$ gives us almost no outliers, see Figure 3.20 and 3.21.

3.4 Cubic Bézier curves with small λ-residuals

In this section, we study the set of cubic Bézier curves that have $e_\lambda < 0.2$. From the above discussion, we take the curves to be visually close to the elastic curves (small L^2 and H^1 distance to an elastica). To study the set, we randomly generate a set $\Lambda_{0.2}$ of cubic Bézier curves in $\mathbb{R}^2$ with $e_\lambda < 0.2$. We fix the endpoints at $(0,0)$ and $(1,0)$, imposing a bound on the outer edge lengths $L_1, L_2 < 2$. We posit that most curves in a design context satisfy $L_1, L_2 < 2$. With this limitation we obtain $\Lambda_{0.2}$ as a four-dimensional set, where each $(x_1, y_1, x_2, y_2) \in \Lambda_{0.2}$ is a cubic Bézier curve with inner control points (x_1, y_1), (x_2, y_2), and $e_\lambda < 0.2$. After generating $\Lambda_{0.2}$, we visualize it as an animation. That is for a discrete set of x_1 we plot the points

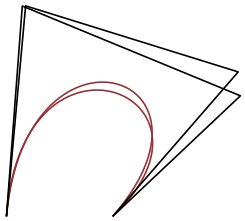


Fig. 3.22.: The curves in $\Lambda_{0.2}$ with the maximal nearest neighbor distance ($= 0.23$). The distance is the Euclidean distance between the inner control points.

$$\{(y_1, x_2, y_2) \,|\, (x_1, y_1, x_2, y_2) \in \Lambda_{0.2}\} \subseteq \mathbb{R}^3 \,,$$

[2] Recall the endpoints are at $(0,0)$ and $(1,0)$.
[3] We considered the distances: $\|\theta - \theta_e\|_\infty$, $\|\gamma - \gamma_e\|_\infty / L$, and the L^2 distance.

where $\in$ in this context means inside of $\Lambda_{0.2}$ with a small tolerance (otherwise the set is empty because $\Lambda_{0.2}$ is a discrete set). In Figure 3.23 we include a subset of these plots. From Figure 3.23 and the animation, we observe that $\Lambda_{0.2}$ contains two separate symmetric regions with the same set of curves but reflected across the x-axis. We also observe that $\Lambda_{0.2}$ appears to be reasonably connected, and hence it makes sense to consider the number of connected components. The number of connected components is important when we later design surfaces foliated by elastica-like Bézier curves. If the set is connected, then we can connect any set of cubic Bézier curves, that have $e_\lambda < 0.2$, with other curves satisfying $e_\lambda < 0.2$, and obtain an interpolating surface foliated by elastica-like Bézier curves. To determine the number of connected components of the set, an examination of the λ-residual is essential. As the λ-residual is given by a rather complicated formula (2.3), we have not yet obtained a formal proof of the number of connected components. Instead, we leave it as future work. In Figure 3.22 we show the two curves with maximal nearest neighbor distance in $\Lambda_{0.2}$.

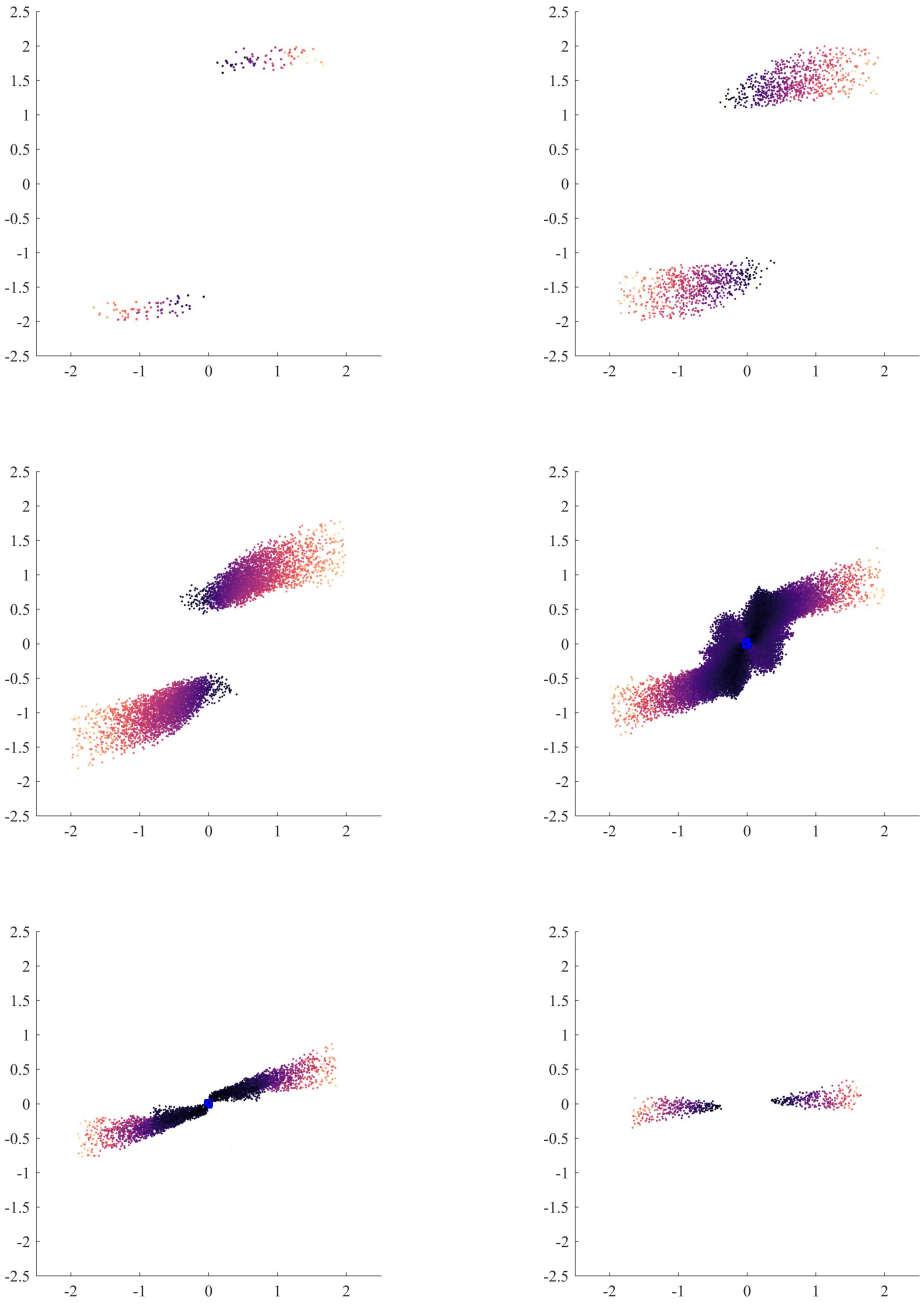


Fig. 3.23.: The plots for $x_1 \in \{-10/9, -6/9, -2/9, 2/9, 6/9, 10/9\}$ from top left to bottom right. The x-axis is the y_1 value and the y-axis is the y_2 value. The color gives us the x_2 value. The blue points denote the lines.

4 Projection to elastica-like curves

In this chapter we introduce a projection tool for cubic Bézier curves. The projection tool takes a cubic Bézier curve as an input and returns an elastica-like Bézier curve, sharing the same endpoints and end tangent angles of the input. The projection tool is a useful design tool. It is a tool that provides an immediate elastica-like alternative if the input is not elastica-like, and it serves as a good compromise between approximating curves with elastica and designing within the space of elastica-like Bézier curves.

One might argue that we can obtain a simple and reliable projection tool by taking a gradient-driven approach, minimizing the bending energy subject to the given constraints. However, this approach has several problems. If we just minimize the bending energy $\frac{1}{2}\int_0^L \kappa^2 \,\mathrm{d}s$, then we observe that the curve length explodes, see Figure 4.1 left. We can try to rectify this by making the bending energy scale invariant, i.e., by considering the energy $\frac{L}{2}\int_0^L \kappa^2 \,\mathrm{d}s$ but this choice also leads to outputs that are not visually close to the elastica, see Figure 4.1 right.

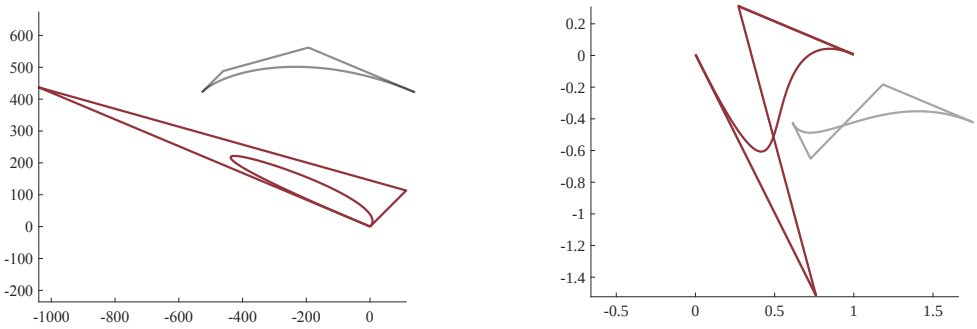


Fig. 4.1.: Left: we minimize the bending energy of the input curve (gray) and obtain the solution (red). Right: we minimize the scale invariant bending energy.

A different candidate for an energy is the λ-residual. In Chapter 3 we observed that a small e_λ gives us a small L^2 distance to the best approximating elastic curve. Hence by minimizing e_λ we get a curve visually close to an elastica. Even though this approach provides us with closeness to an elastica, there is still a problem of unreliability. Meaning that we might have two nearly identical curves that project to two very different elastica-like Bézier curves. An example of this is shown in Figure 4.2. Consequently, we must take another approach. The approach, that we use, takes advantage of the five constraints for closeness determined in Section 3.2.3.

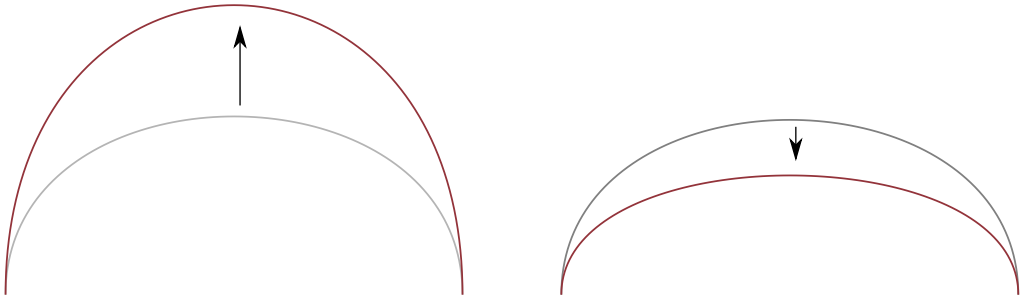


Fig. 4.2.: Minimizing the λ-residual might lead to unreliable inputs (the gray curves). The nearly identical input curves project to two very different outputs (red curves).

4.1 Projection: fixed endpoints and end tangent angles

Given an input curve we assume Angle Constraints 1–2 are satisfied. If not, we satisfy Angle Constraints 1–2 by rotating the outer edges in the control polygon (but this of course changes the end tangent angles). With the angle constraints satisfied, we then make sure that the curve does not have a self-intersecting polygon and meets the length constraints . We satisfy these constraints by moving the inner control points up or down on the outer edges of the control polygon. By shortening or extending the outer edges we do not change the endpoints or end tangent angles. Having fulfilled the length and angle constraints, we only need to move (ϕ_1, ϕ_2) to the projection zone.

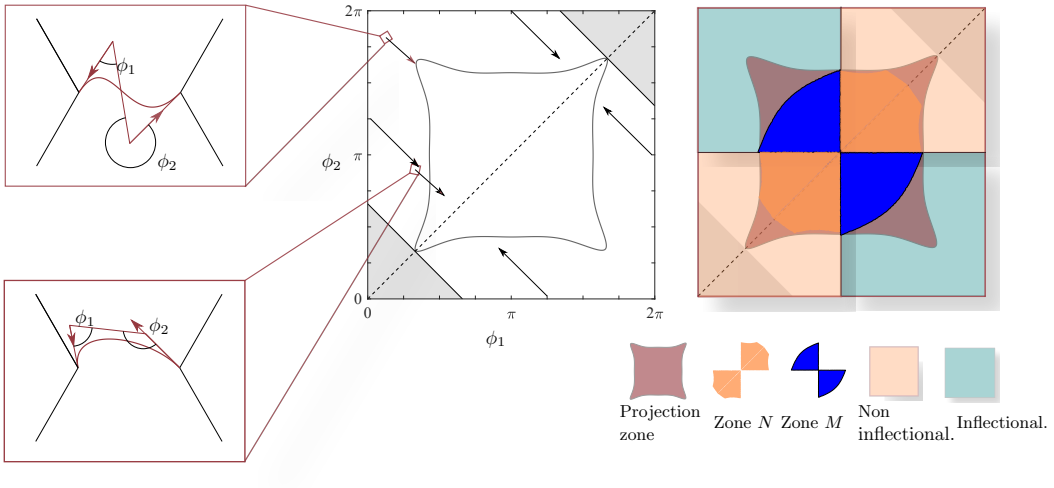


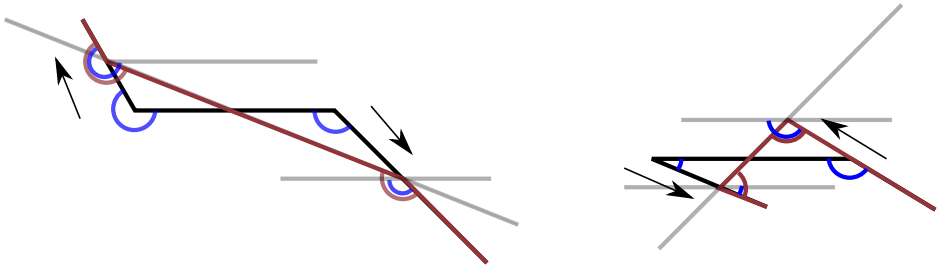
Fig. 4.3.: The projection tool that preserves endpoints and end tangent angles.

We move to this zone by moving the inner control points on the outer edges such that we approach the line $\phi_1 = \phi_2$ in the middle of the zone, see Figure 4.3. To move (ϕ_1, ϕ_2) of the input curve to this line, we consider two different cases: the inflectional and non-inflectional case. The inflectional and non-inflectional case each

have a region in the (ϕ_1, ϕ_2)-plane. Figure 4.3 shows these regions as respectively green and light orange.

4.1.1 Inflectional case

An inflectional curve either has $\pi \leq \phi_1, \phi_2 \leq \pi$ or $\phi_1 \leq \pi, \pi \leq \phi_2$. For an inflectional curve, we observe that if we shorten the outer edges of the control polygon, then ϕ_1 and ϕ_2 become more equal, i.e., the small angle becomes larger and the large angle smaller. This is illustrated in Figure 4.4a.



(a) The inflectional case. **(b)** The non-inflectional case.

Fig. 4.4.: After shortening/extending the outer edges of the input control polygon (black) the angles (red) become more equal.

To move (ϕ_1, ϕ_2) to $\phi_1 = \phi_2$ we therefore move the lengths of the outer edges to $L_{\min}$ given by (3.1). We adjust L_1 and L_2 according to the scheme

$$L_1(t) = (1-t)L_1 + tL_{\min}, \quad L_2(t) = (1-t)L_2 + tL_{\min}, \tag{4.1}$$

with $t \in [0, 1]$. Because $\phi_1 + \phi_2 = \beta_1 + \beta_2$ and β_1 and β_2 are fixed under (4.1), the sum $\phi_1 + \phi_2 = c$ stays constant when we apply the scheme. Hence we move along the line towards the point $(c/2, c/2)$ as illustrated in Figure 4.3. If the curve becomes non-inflectional we apply the algorithm described below for the non-inflectional case. Otherwise, we reach the projection zone with this scheme because the curves with $L_1 = L_2 = L_{\min}$ are all contained inside of the projection zone. In Figure 4.3 we have plotted the region $L_1 = L_2 = L_{\min}$ as the blue Zone M. Once we reach the projection zone we return the output as the projected curve. The projected curve is elastica-like because we do not introduce a self-intersecting control polygon and it satisfies the criterion for being elastica-like. To remove any doubt, we list and verify the five constraints here:

- ✓ Angle Constraint 1–2 are satisfied by the input and never violated by (4.1).
- ✓ Lengths Constraint 1–2 are satisfied. Length Constraint 1 is satisfied after adjusting the input and never violated by (4.1). Length Constraint 2 is satisfied

because we move (L_1, L_2) to $(L_{\min}, L_{\min})$ along a straight line (and hence the fraction becomes closer to 1).

✓ The (ϕ_1, ϕ_2) Constraint is satisfied by the stopping criterion.

4.1.2 Non-inflectional case

For non-inflectional curves we have the following four cases: $\phi_1 \leq \phi_2 \leq \pi$, $\phi_2 < \phi_1 \leq \pi$, $\pi \leq \phi_1 \leq \phi_2$, and $\pi \leq \phi_2 < \phi_1$. Assuming without loss of generality that $\phi_1 \leq \phi_2 \leq \pi$, we observe that we project towards $\phi_1 = \phi_2$ by decreasing L_1 and increasing L_2, see Figure 4.4b. The scheme we use is therefore similar to (4.1):

$$L_1(t) = (1-t)L_1 + tL_{\min}, \quad L_2(t) = (1-t)L_2 + tL_{\max}, \tag{4.2}$$

with $t \in [0, 1]$. We have plotted the region $L_1 = L_{\min}$ and $L_2 = L_{\max}$ in Figure 4.3 as the orange zone N. Because N is contained in the projection zone, we are guaranteed to reach the projection zone before violating Length Constraint 1. Again, we do not introduce a self-intersecting control polygon, violate the length constraints, and Angle Constraints 1–2 while applying (4.2), and hence the output is elastica-like.

In Figure 4.5 we include examples of input and output of the proposed projection tool. For these examples, we also plot the initial guesses of the approximating elastic curves. The initial guess is only an estimate of the best approximating elastic curve and we can make it closer to the output by applying an optimization routine.

4.1.3 Feedback projection

We obtain a feedback based projection tool by utilizing that the λ-residual can be computed at interactive speeds while adjusting L_1 and L_2 with (4.1) or (4.2). If e_λ becomes below a given threshold E_λ, we terminate the procedure and return the curve. In addition to computing the λ-residual, we also choose to adjust L_1 and L_2 to a given set of local minimizers for e_λ. For inflectional curves, local minimizers of e_λ appear to be close to satisfying $L_1 = L_2 = L_{\min}$. Hence we do not replace the scheme (4.1). However, for (4.2) we move L_1 and L_2 towards new values $\mathcal{L}_1$ and $\mathcal{L}_2$. For L_1 we use

$$\mathcal{L}_1(\theta_1, \theta_2) = \min(0.12, f(\theta_1, \theta_2)),$$

where

$$\begin{aligned} f(\theta_1, \theta_2) =& 0.00001475 \exp(10.39\theta_1 - 10.48\theta_2) + 0.4574\theta_1 \exp(1.711\theta_1 - 2.535\theta_2) \\ &+ 2.772\theta_2 \exp(-0.08504\theta_1 - 0.9109\theta_2) - 0.2957\theta_1\theta_2 \exp(-0.6606\theta_1), \end{aligned}$$

and $0 \leq \theta_1$ and θ_2 denote the angles from the positive x-axis to the outer edges. By symmetry we obtain formulas for $\theta_1 \leq 0$ and $\mathcal{L}_2$. We obtained the function f by computing a set of (θ_1, θ_2) for non-inflectional curves, satisfying Angle Constraints 1–2, and for each (θ_1, θ_2) we determined the local minimizers $\mathcal{L}_1$ and $\mathcal{L}_2$ with e_λ as the energy. Using the data $(\theta_1, \theta_2, \mathcal{L}_1)$ we then made a 3-D plot that we fitted with the function f, giving us a small sum of squares error.

With these changes to the projection tool, we summarize the feedback projection here:

1. For a given input curve we transform the curve to have endpoints at $(0, 0)$ and $(1, 0)$. We assume the input already satisfies Angle Constraints 1–2.

2. We remove any self-intersections in the polygon by adjusting L_1 and L_2.

3. We adjust L_1 and L_2 to $[L_{\min}, L_{\max}]$ and determine if the curve is inflectional or non-inflectional.

4a. For an inflectional curve we apply (4.1) to the adjusted curve until $L_1 = L_2 = L_{\min}$, $e_\lambda \leq E_\lambda$, or the curve becomes non-inflectional (in this case we apply 4b. below).

4b. For a non-inflectional curve we use

$$L_1(t) = (1-t)L_1 + t\mathcal{L}_1, \quad L_2(t) = (1-t)L_2 + t\mathcal{L}_2$$

until $L_1 = \mathcal{L}_1$ and $L_2 = \mathcal{L}_2$ or $e_\lambda \leq E_\lambda$.

5. We apply the inverse transformation and obtain the projected curve.

To test the feedback algorithm, we have considered a sample of 100,000 random cubic Bézier curves that all satisfied Angle Constraints 1–2. We applied the algorithm with $E_\lambda = 0$ and $t \in \{0, 0.1, \ldots, 0.9, 1\}$. For the smallest value of e_λ we then reapplied the algorithm in a neighborhood of the respective t value. From the output we obtained mean(e_λ)=0.048, median$(e_\lambda) = 0.042$, and $\max(e_\lambda) = 0.26$.

We also considered the problem of projecting curves that do not satisfy Angle Constraints 1–2. We computed 83,500 random Bézier curves and out of these 45% violated Angle Constraints 1–2. For these curves, we got a mean, median, and max of respectively $0.4, 0.3$, and (approximately $\sim$) 1. Hence we conclude that we cannot remove the angle constraints. Instead of removing the angle constraints, we have therefore considered a relaxation of the angle constraints. With a relaxation, we observe from Table 4.1 that we still obtain a useful projection tool.

Tab. 4.1.: The results of the feedback projection algorithm with different angle constraints.

Angle Constraint 1	Angle Constraint 2	Mean(e_λ)	Median	Max
$\beta_1, \beta_2 \in (\pi/3, 2\pi - \pi/3)$	$\lvert\beta_1 - \beta_2\rvert < 0.4\pi$	0.048	0.042	0.26
No constraint	No constraint	0.4	0.3	~ 1
$(\pi/4, 2\pi - \pi/4)$	0.6π	0.06	0.043	0.42
$(\pi/6, 2\pi - \pi/6)$	0.75π	0.1	0.06	0.71

In Figure 4.7 we include different examples of input and output of the feedback projection algorithm with $E_\lambda = 0$. In Figure 4.6 we demonstrate the impact of E_λ for three different input curves (black dashed).

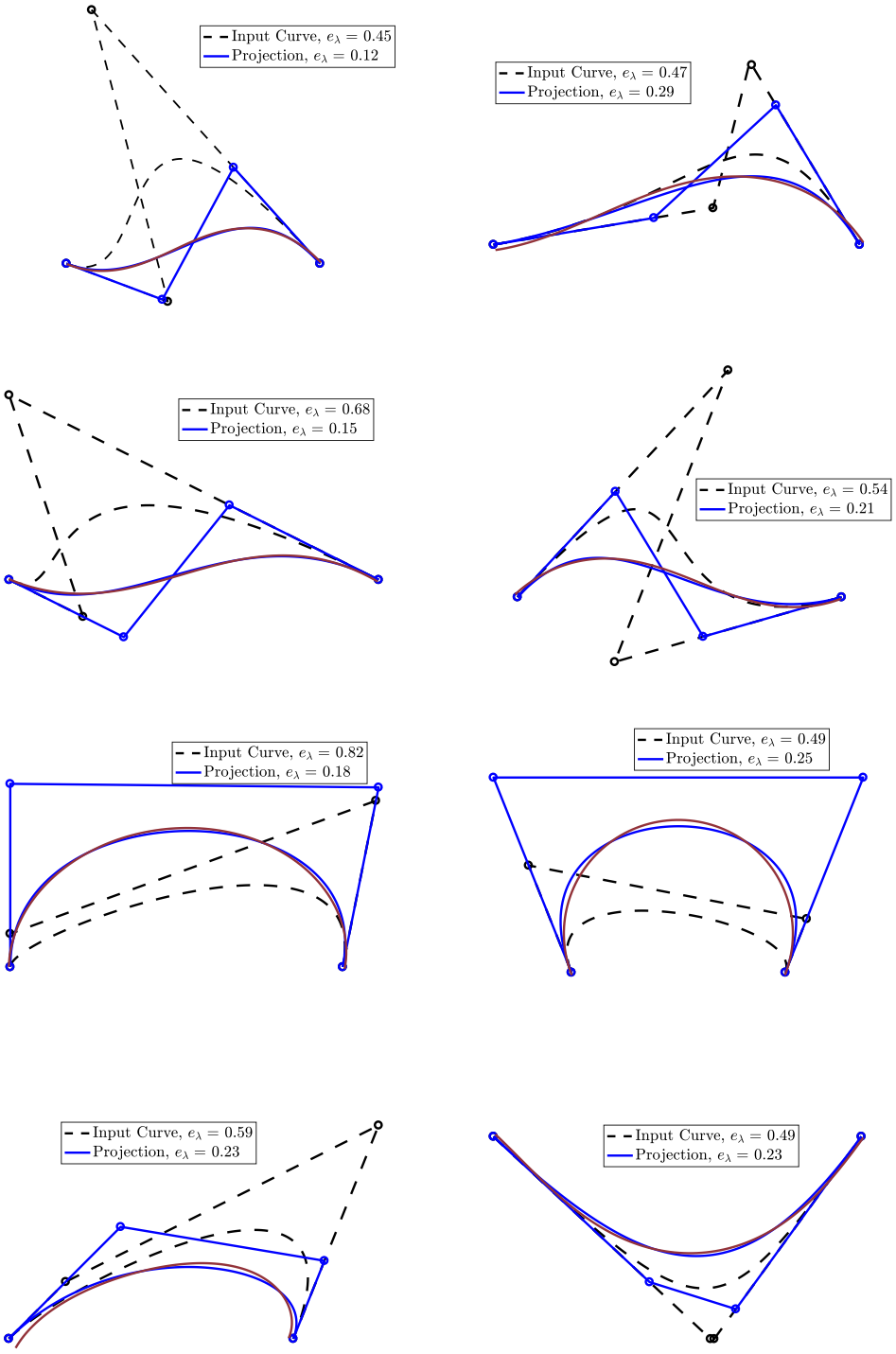


Fig. 4.5.: The projection algorithm applied to eight curves (black dashed). The blue curve is the projected output and the red curve is the initial guess of an approximating elastica. We can get a better elastica approximation by applying an optimization routine.

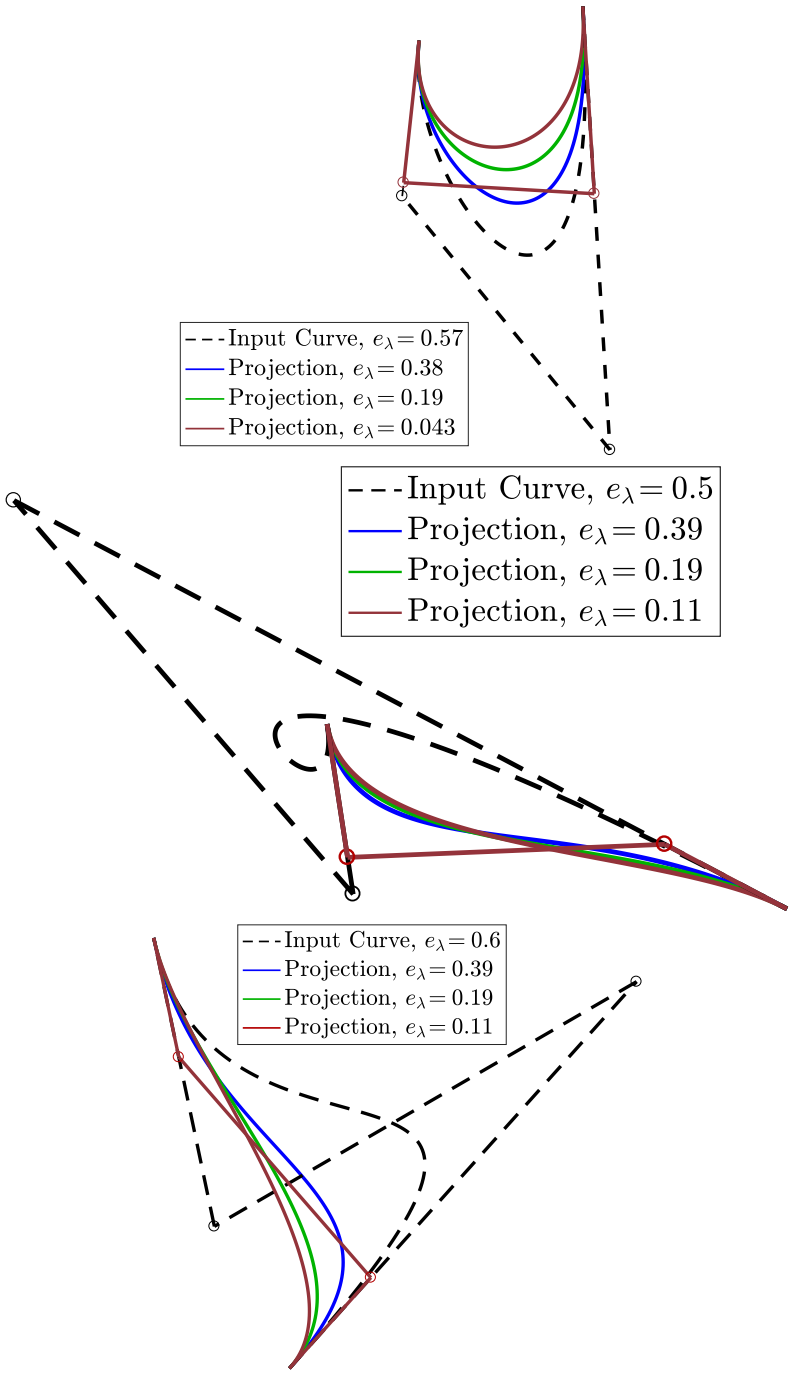


Fig. 4.6.: The feedback projection algorithm with different thresholds $E_\lambda \in \{0, 0.2, 0.4\}$.

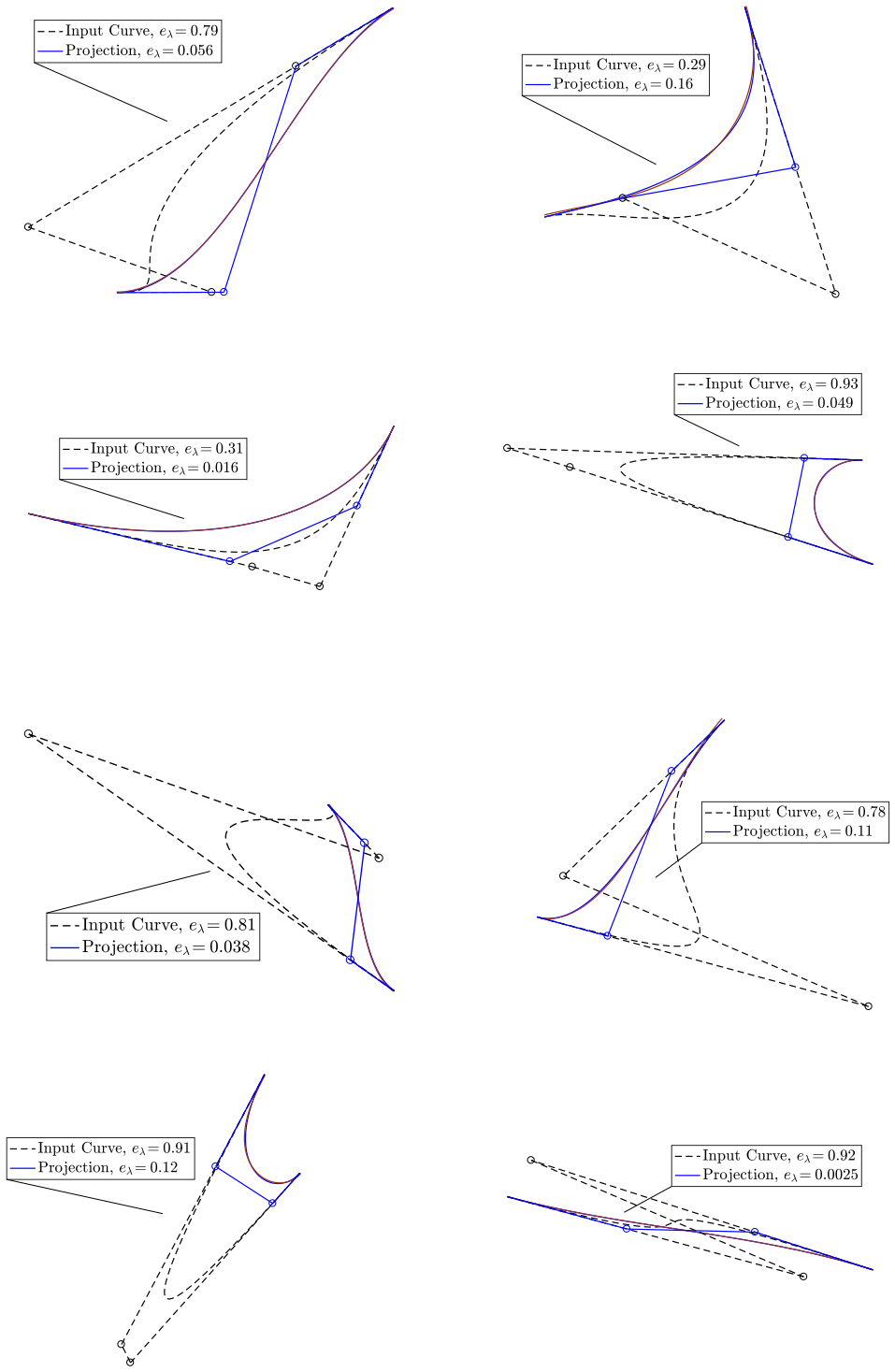


Fig. 4.7.: The feedback projection algorithm applied to eight curves (black dashed) with $E_\lambda = 0$. The blue curve is the projected output and the red curve is the initial guess of an approximating elastica (they coincide).

5 Design of elastic splines

Physical or elastic splines can be traced back through history. In the stone age, humans used physical splines to create shelters. The physical splines were made by attaching and bending several pieces of wooden strips. From these wooden strips, the shelter got a round shape, which was then covered in animal skins to protect against the wind. Physical splines were also used in shipbuilding. An early example of this is the ship Culip VI that can be dated back to the late 13th century AD. This ship was created by wooden ribs that were shaped by a wooden template [Rie03]. Later this technique became more refined, and the ship designs were carried out on paper using a physical spline and ducks. By bending the spline and holding it in place by the ducks, the ship engineer could then draw the curves needed for the ship design. Today curves are often drawn in CAD systems using polynomial splines that are piecewise Bézier curves. A common choice is to design with cubic splines but, as mentioned in Chapter 3, not all cubic Bézier curves visually represent an elastic curve, and hence we cannot expect a cubic spline to be close to an elastic spline. Consequently, we need a different tool for designing elastic splines in a CAD environment.

In the literature different methods for representing an *interpolating* elastic/non-linear spline on a computer have been considered [Gla66; Woo69; Meh74; Mal77; Edw92; BK94]. The articles [Gla66] and [Woo69] describe algorithms that determine a non-exact C^2 elastic spline, which interpolates a given set of points. The starting point of the algorithms is a fourth-order differential equation, obtained by calculus of variations, that is solved numerically. A different approach is introduced in [Meh74], where the author obtains a non-exact elastic spline with the KURGLA (KURveGLAtting[1]) algorithms. KURGLA I uses circular arcs to represent the non-exact elastic spline, i.e., the algorithm approximates the curvature function of the elastic spline by a piecewise constant function. The other algorithm, KURGLA II, uses a piecewise linear function to approximate the curvature. This way the non-exact elastic spline consists of pieces of the Euler spiral. The article [Mal77] discusses the algorithms in [Gla66; Woo69; Meh74], and presents a finite difference method for obtaining the non-exact elastic spline. Contrary to the algorithms in [Gla66; Woo69; Meh74; Mal77], the article [Edw92] proposes a method that obtains an interpolating elastic spline by means of a basis function. The basis function is obtained from the Euler equation of an elastic curve (with unconstrained length):

[1] Norwegian for curve smoothing.

$(\mathrm{d}^2\kappa/\mathrm{d}s^2) + (1/2)\kappa^3 = 0$. From this function, the author derives non-linear equations describing a pointwise interpolating C^2 elastic spline. To solve the equations (and get the elastic spline) the author applies Newton's method. One of the more recent articles [BK94] presents an algorithm similar to [Meh74]. The algorithm in [BK94] obtains a non-exact interpolating elastic spline. The non-exact elastic spline is a C^1 curve with a piecewise polynomial curvature function and little curvature discontinuity.

In this chapter we provide an alternative method to the existing methods in the literature. By applying the basic idea of Method 2 from Chapter 2, we use a cubic spline as an interpolating non-exact elastic spline with C^1 or C^2 continuity. Compared to the methods in the literature, the cubic spline representation gives us a very intuitive design curve for creating a non-exact elastic spline. The cubic spline representation fits well into the framework of most CAD systems/NURBS libraries. However, the method is not interactive[2]. This method and the existing methods are not interactive for two reasons. First, we lack shape control and robustness because we specify the shape through a set of interpolating values and there is not in general a unique solution. Secondly, we must apply an optimization routine/iteration scheme, and this can be time expensive and cause unexpected output from the user's input. To circumvent these problems, we present in Section 5.2, a data-driven method for interactively designing elastic splines. In the last section of this chapter, we also consider the problem of designing surfaces foliated by elastic splines. In particular, we explain how we use the spline tools to create production-ready designs for the hot-blade technology.

5.1 Design of elastic splines via interpolation and optimization

We use the idea of Method 2, Chapter 2, to obtain numerical C^1 and C^2 elastic splines that interpolate a sequence of points and tangent angles at the points. We propose a two-step method:

- first, we construct an initial C^1 or C^2 cubic spline that interpolates the given input values, and then
- we repeatedly minimize the bending energy and insert knots. We minimize the bending energy subject to the interpolation constraints and curve length of the curve segments/spline.

[2] Allows direct communication between user and output.

Below we explain the two steps in details for the C^1 and C^2 constructions.

5.1.1 Numerical C^1 elastic splines

Given a sequence of n interpolation points and n tangent angles at the points (with a predetermined speed), we first compute the initial C^1 cubic spline that interpolates the points and tangents. The initial spline is a succession of $n-1$ cubic Bézier curves that meet with C^1 continuity at the interpolation points with the specified tangents. We obtain this succession by putting conditions on the control polygons for each curve. This is possible because the end tangents and endpoints for each curve are given by a linear equation of the first and last two points in the control polygon of the curve. After having constructed the initial spline, we obtain the numerical C^1 elastic spline by repeatedly minimizing the bending energy and inserting knots. We minimize the bending energy of the spline subject to the constraint that all curve segments have fixed endpoints, end tangent angles, and curve length (given by the initial spline). For more details on the minimization problem, we refer to Method 2 in Section 2.2.2.

5.1.2 Numerical C^2 elastic splines

To obtain numerical C^2 elastic splines, we only specify for our interpolation:

- the n interpolation points $P_1, \dots, P_n$ and
- the far left and right end tangent angles of the spline.

Given the n interpolation points $P_1, \dots, P_n$ and the far left/right end tangent angles with a predetermined speed, i.e., tangents t_1, t_2, we construct an interpolating C^2 cubic spline $\gamma_{c_1,\dots,c_{n+2}}$ as our initial spline. The cubic spline $\gamma_{c_1,\dots,c_{n+2}}$ is given by a set of control points $c_i \in \mathbb{R}^2$, B-spline functions B_i of degree 3, and knot vector $\tau = (0,0,0,0,\frac{1}{n-1},\dots,\frac{n-2}{n-1},1,1,1,1)$:

$$\gamma_{c_1,\dots,c_{n+2}}(t) = \sum_{i=1}^{n+2} c_i B_i(t)$$

such that

$$\gamma'_{c_1,\dots,c_{n+2}}(0) = t_1, \quad \gamma'_{c_1,\dots,c_{n+2}}(1) = t_2, \quad \gamma_{c_1,\dots,c_{n+2}}(0) = P_1, \quad \gamma_{c_1,\dots,c_{n+2}}\left(\frac{1}{n-1}\right) = P_2,$$
$$\dots \quad , \gamma_{c_1,\dots,c_{n+2}}(1) = P_n\,. \tag{5.1}$$

To obtain $\gamma_{c_1,\ldots,c_{n+2}}$ we include four of these constraints directly in the parameterization of the spline by means of the equations:

$$c_1 = P_1, \quad c_2 = \frac{1}{3(n-1)}t_1 + P_1, \quad c_{n+1} = P_n - \frac{1}{3(n-1)}t_2, \quad c_{n+2} = P_n.$$

Having specified these we only need to determine the control points $c_3, \ldots, c_n$. We obtain the control points from the remaining $n-2$ constraints:

$$\begin{aligned}
\gamma_{c_1,\ldots,c_{n+2}}\left(\frac{1}{n-1}\right) &= \sum_{i=1}^{n+2} c_i B_i\left(\frac{1}{n-1}\right), \\
&\vdots \\
\gamma_{c_1,\ldots,c_{n+2}}\left(\frac{n-2}{n-1}\right) &= \sum_{i=1}^{n+2} c_i B_i\left(\frac{n-2}{n-1}\right).
\end{aligned}$$

Because the equations are linear in the control points, we obtain a linear system, which we solve to obtain the initial guess for the curvature continuous numerical elastic spline:

$$\begin{pmatrix} B_3(\frac{1}{n-1}) & \ldots & B_n(\frac{1}{n-1}) \\ \vdots & \ldots & \vdots \\ B_3(\frac{n-2}{n-1}) & \ldots & B_n(\frac{n-2}{n-1}) \end{pmatrix} \begin{pmatrix} c_3 \\ \vdots \\ c_n \end{pmatrix} = \begin{pmatrix} P_2 - c_1 B_1(\frac{1}{n-1}) - c_2 B_2(\frac{1}{n-1}) - c_{n+1} B_{n+1}(\frac{1}{n-1}) - c_{n+2} B_{n+2}(\frac{1}{n-1}) \\ \vdots \\ P_{n-1} - c_1 B_1(\frac{n-2}{n-1}) - c_2 B_2(\frac{n-2}{n-1}) - c_{n+1} B_{n+1}(\frac{n-2}{n-1}) - c_{n+2} B_{n+2}(\frac{n-2}{n-1}) \end{pmatrix}.$$

As we have a uniform knot vector and evaluate at the knots, the matrix has a simple structure:

$$\begin{pmatrix} \frac{7}{12} & \frac{2}{12} & & & & & \\ \frac{2}{12} & \frac{8}{12} & \frac{2}{12} & & & & \\ & \frac{2}{12} & \frac{8}{12} & \frac{2}{12} & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & & \frac{2}{12} & \frac{8}{12} & \frac{2}{12} \\ & & & & & \frac{2}{12} & \frac{7}{12} \end{pmatrix} \begin{pmatrix} c_3 \\ \vdots \\ c_n \end{pmatrix} = \begin{pmatrix} P_2 - c_2 \frac{3}{12} \\ P_3 \\ \vdots \\ P_{n-2} \\ P_{n-1} - c_{n+1} \frac{3}{12} \end{pmatrix}.$$

After having solved this system, we obtain the numerical C^2 elastic spline by repeatedly minimizing the bending energy and inserting knots. In this case we minimize the bending energy subject to the interpolation points, two end tangent angles, and the constraint that the total length of the spline is fixed.

5.1.3 Results

We have tested the numerical C^1 and C^2 elastic spline algorithms with different inputs. Examples are shown in Figures 5.1 and 5.2. Even though the C^1 and C^2 interpolation tools provide the designer with enough design freedom to create a variety of (numerical) elastic splines, there are still limitations.

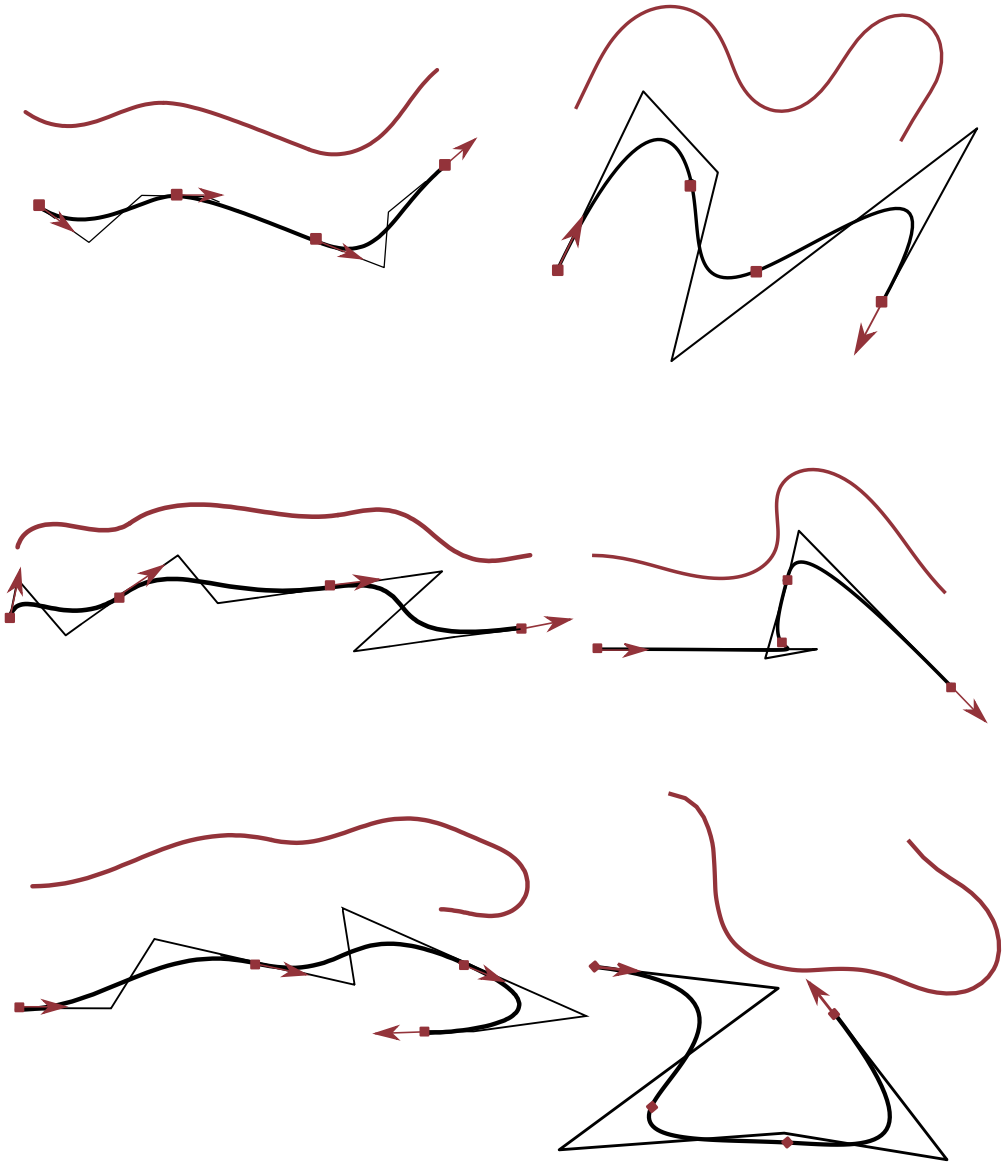


Fig. 5.1.: Black: the input spline. Red: numerical elastic spline with C^1 continuity (3 segments). The red arrows and points are the interpolation points and tangents.

Fig. 5.2.: Black: the input spline. Red: numerical elastic spline with C^2 continuity (3 segments). The red arrows and points are the interpolation points and tangents.

We lack shape control because elastic curves/splines are not uniquely determined by the boundary values/interpolation. In Figure 5.3 we display different elastic curves,

sharing the same endpoints, end tangents, and length. The non-uniqueness of a solution also means we have a problem of robustness. Sometimes we obtain curves where a slight change in the boundary values, or interpolation, causes the curve to jump configuration, e.g., Figure 5.4. Furthermore, the tools can be time expensive because we apply an optimization routine.

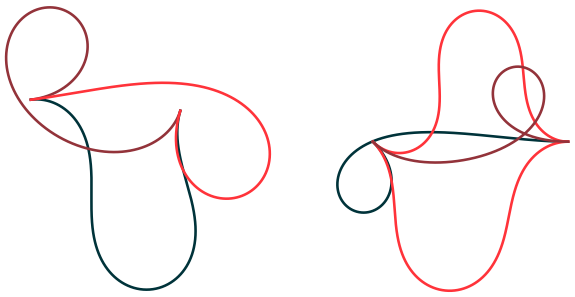


Fig. 5.3.: Two sets of elastic curves with same endpoints, end tangents, and lengths. Examples provided by Toke Bjerge Nørbjerg. More examples are provided in [Ard18].

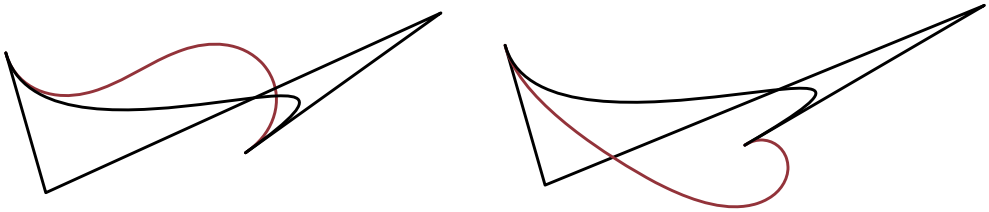


Fig. 5.4.: We extract the boundary conditions from the black cubic Bézier curves and compute the red elastic curves, satisfying these constraints. The small change in the boundary conditions causes the elastic curves to jump configuration.

5.2 Design of elastic splines via a database

In this section, we introduce a data-driven tool for designing elastic splines. As mentioned in the introduction to this chapter, the tool circumvents the limitations of the previously mentioned methods: it is interactive, robust, and we have good shape control of the output. The main idea of the data-driven tool is that we design the elastic spline using a cubic spline proxy/interface where each cubic piece has a small λ-residual (2.3). We let the designer modify the cubic spline, as he/she usually does, but we restrict the position of the control points to *good* regions where the cubic pieces have small λ-residuals. If we restrict the control points to these regions, then we get a cubic spline that takes the approximate shape of an elastic spline. The good regions are computed by means of a database of cubic Bézier curves and their λ-residuals.

The database

The database consists of cubic Bézier curves $\gamma_{c_1,\ldots,c_4}$ given by four control points (c_1, c_2, c_3, c_4). That is $\gamma_{c_1,\ldots,c_4} = \sum_{i=1}^{4} c_i B_{3i}$. In the database we consider only curves in a standard position, meaning that we impose $c_3 = (0, 1)$ and $c_4 = (0, 0)$. With this position, each cubic curve is uniquely determined by four coordinates $c_1, c_2 \in \mathbb{R}^2$ or $c_0 = 2c_1 - c_2, c_2 \in \mathbb{R}^2$, see Figure 5.5. We choose to create the database storing the control point c_2 and the point c_0. In the database we also store the λ-residual e_λ and the angle error θ_{err} of each curve. The angle error is the maximal difference between the end tangent angles of the initial guess, defined in Section 2.1.1, and the cubic curve. The angle error is essential when we concatenate cubic curves to define an elastic spline. In this case we want to obtain a tangent continuous elastic spline and consequently the end tangent angles of each Bézier curve cannot deviate too much from the end tangent angles of the elastica they represent. We create the database from two sets of cubic curves. Using c_1 we sample c_2 and vice versa. We obtain the sets by placing a uniform grid over the left two quadrants and a uniform grid covering all of the quadrants, see Figure 5.5. For each $i = 1, 2$, we sample c_i in the left two quadrants and the other point $c_j, j \neq i$, in the larger grid. Given a position c_i we compute $\|\nabla e_\lambda\|$ for the curves with c_j in the larger grid. Based on the 25th, 50th, and 75th - percentiles of $\|\nabla e_\lambda\|$ we update the uniform grid for c_j such that we obtain more curves when e_λ is rapidly varying. From the position of c_i and the updated grid we obtain a set of curves and transfer the data c_0, c_2, e_λ, and θ_{err} to a database. In Figure 5.6 we show how we use $\|\nabla e_\lambda\|$ to update the uniform grid.

The size of the database depends on the size and density of the grids. We are able to reduce the size of the database by only considering curves with e_λ and θ_{err} below specified tolerances. We used the tolerances $e_\lambda < 0.21$ and θ_{err} less than 6 degrees. In total we obtained about 385,000 curves where for each curve we specified the tuple $(c_0, c_2, e_\lambda, \theta_{\text{err}})$, i.e., six real numbers.

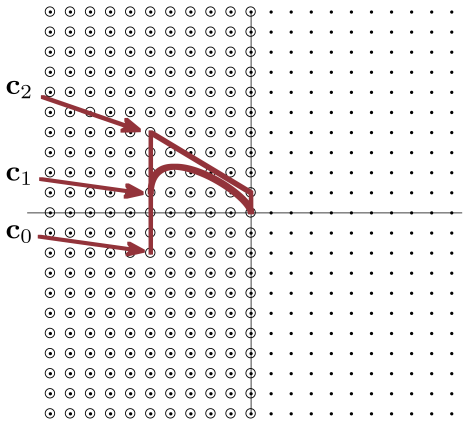


Fig. 5.5.: The uniform grids used to sample the data.

The design tool

Having obtained the database, we introduce the data-driven design tool. Given that we want to design a C^1 elastic spline with n segments we use a C^1 cubic spline proxy γ with $2(n + 1)$ control points on a uniform knot vector, where each cubic piece is constrained to have a small λ-residual and hence is close to an elastica. In the design tool, we first initialize γ to be close to an elastic spline, e.g., a line.

Fig. 5.6.: Left: plot of positions for c_2 where the color indicates the value of e_λ. The black dashed curves are positions where the curve has a cusp. Center: the updated grid for c_2. Right: plot of c_2 positions where the color indicates $\|\nabla e_\lambda\|$.

After this, the designer chooses a control point as an active point: the point that he/she wants to reposition and that modifies the spline. Given this active point, we determine the good region: the region where the active point can be placed such that γ continues to be close to an elastic spline. Figure 5.7 shows the good region, the active point, and the cubic spline. To obtain the good region we apply the following steps:

1. Given the active point, we transform (scale, rotate, and translate) the cubic curves, affected by the active point, to the standard position. That is the last two control points, unaffected by the active point, transform to $(0, 1)$ and $(0, 0)$. Because we impose C^1 continuity we affect up to two cubic Bézier curves.

2. For each affected curve we consider the position of the *fixed point* and determine the good region with respect to the curve. The fixed point is the control point or c_0 point that does not move when the active point is put in a new position. From the position of the fixed point, we determine the good region by searching the database for elastica-like curves with a point that matches the fixed point. From the matches in the database, we extract the positions of the c_0 or control point that corresponds to the position of the active point. This set of positions gives us a region where the active point can be placed such that the affected curve is close to an elastica. Below, we explain this step with an example.

3. Having determined the good region for each affected curve, we compute the good region of the spline as the intersection of these sets.

In the general case, two curves are affected by the active point, e.g., Bézier curve 1–2 in Figure 5.7. For Bézier curve 1 we obtain the transformed curve in Figure 5.8.

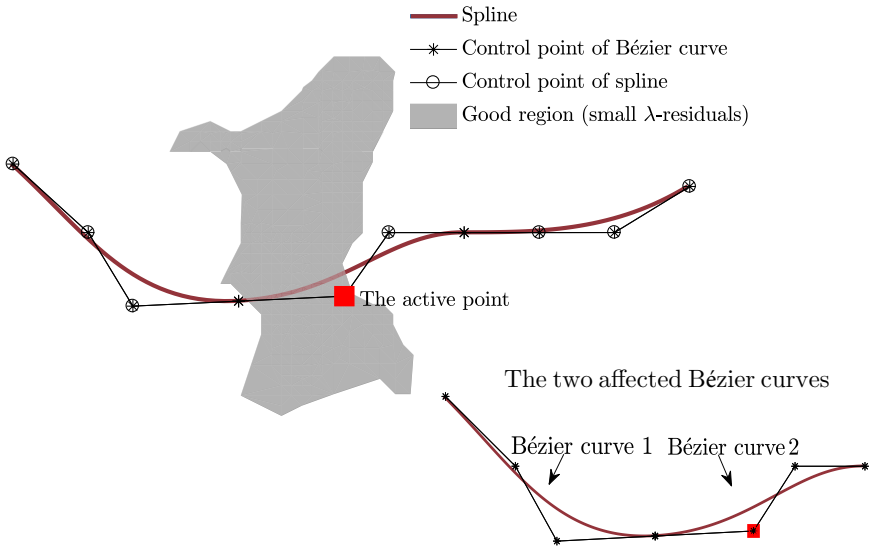


Fig. 5.7.: The cubic spline with the active point.

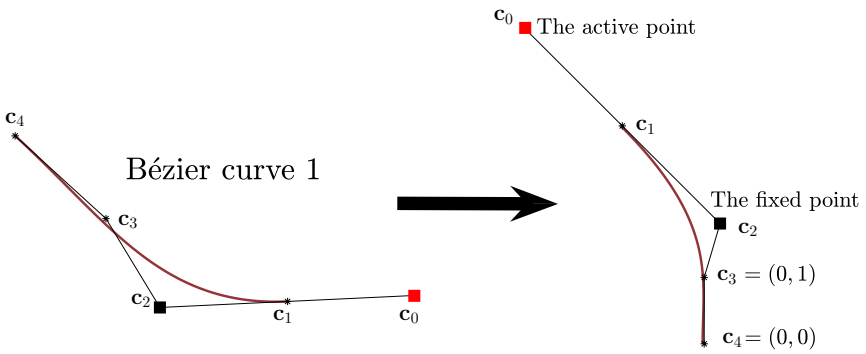


Fig. 5.8.: The transformation of Bézier curve 1 from Figure 5.7.

In this case the fixed point is the c_2 point, and we determine the set of $(c_0, c_2, e_\lambda, \theta_{\text{err}})$ in the database with a c_2 point close to the c_2 point of the transformed Bézier curve 1. From the set of matches, we retrieve the c_0 positions, which we transform using the inverse transformation. This set of points is the good region for Bézier curve 1. For Bézier curve 2 the idea is the same. We transform the curve to the standard position, as shown in Figure 5.9, and the fixed point is now the c_0 point, which we compare to the c_0 point of the curves in the database. Having determined the matches $(c_0, c_2, e_\lambda, \theta_{\text{err}})$, we apply the inverse transformation to the c_2 points and obtain the good region for Bézier curve 2. Thus we have for Bézier curve 1–2 a respective good region, and we compute the good region of the spline as the intersection of these sets. We visualize the intersection as an α-shape [Ede+83]. Having determined and visualized the good region, the designer is now free to move the active point inside this region.

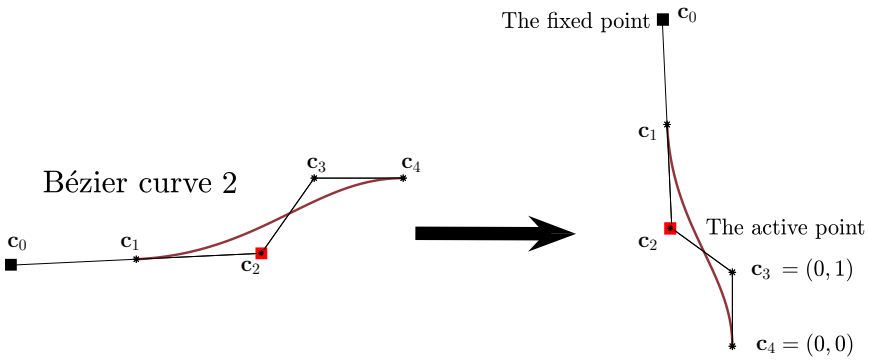


Fig. 5.9.: The transformation of Bézier curve 2 from Figure 5.7.

This way the spline stays close to an elastic spline, and the designer can choose a new active point to repeat the procedure and continue designing the elastic spline. Because step 1–3 can be computed at interactive speeds, the tool is an interactive design tool for designing C^1 elastic splines. Furthermore, this tool generalizes to C^2 elastic splines, but the procedure gets more complicated, because the active point affects up to four polynomial pieces.

5.2.1 Block segmentation

In Figures 5.10 to 5.12, we demonstrate the design tool. In the demonstration we also include a block segmentation (the dashed lines). That is, we show where we want the elastic curve segments to start and end. A block segmentation is necessary if we want to use the tool in the context of hot-blade cutting, e.g, if we want to use a hot-blade robot to cut a design from multiple EPS blocks. If we just apply the above tool, we might violate a given segmentation after moving the active point. We can rectify the violation by moving the neighboring point to the active point. We move the point as little as possible by moving it orthogonal to the block boundary. After this, the resulting cubic spline might deviate from being an elastic spline. To guarantee closeness after updating the neighboring point, we therefore suggest to compute the good region with a predetermined block segmentation in mind. For instance, in Figure 5.13, we use the axes e and f to compute the positions for the active point such that Bézier curve 2 remains close to an elastica after placing the active point *and* updating the neighboring point c_0. The axes e and f are respectively orthogonal and parallel to the block boundary.

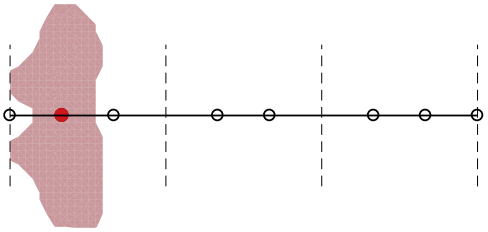


Fig. 5.10.: The red region is the good region for the active point.

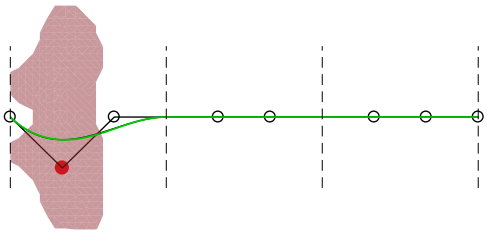


Fig. 5.11.: After we reposition the active point.

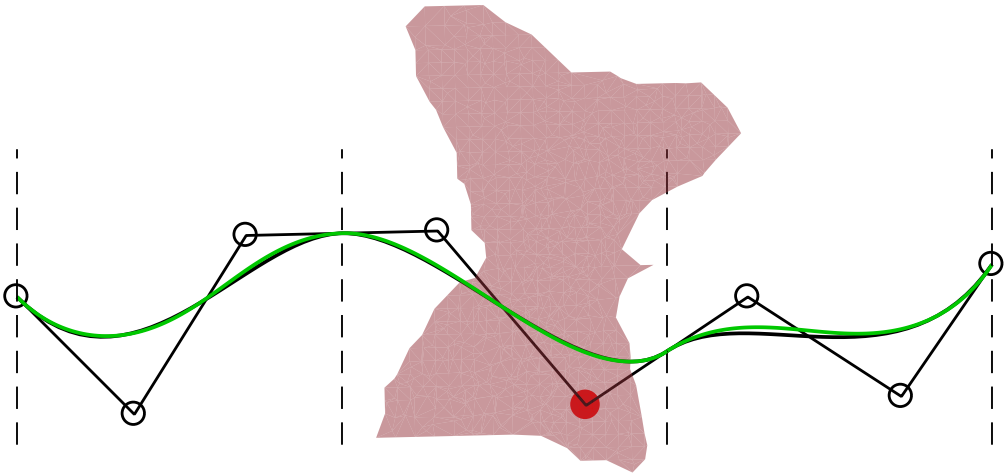


Fig. 5.12.: Final curve after several iterations of moving the control points to good regions. The green curve is a C^1 concatenation of the initial guesses.

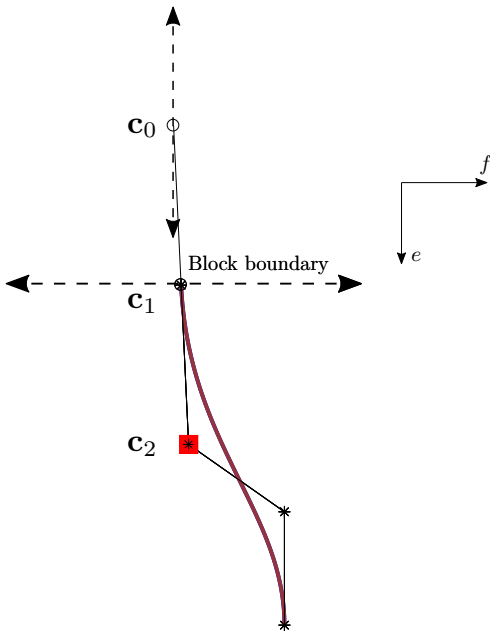


Fig. 5.13.: Bézier curve 2, the block segmentation, axes e and f.

Using e and f we search the database for $(c_0, c_2, e_\lambda, \theta_{\text{err}})$ such that

$$|(c_0 - \hat{c}_0) \cdot f| < \varepsilon, \quad |(c_1 - \hat{c}_1) \cdot e| < \varepsilon\,, \tag{5.2}$$

with $\varepsilon > 0$ being a small tolerance and where $\hat{c}_0, \hat{c}_1$ denote the c_0 and c_1 point of Bézier curve 2. From this, we obtain a set of tuples $(c_0, c_2, e_\lambda, \theta_{\text{err}})$ satisfying (5.2) and from these we get the region for Bézier curve 2 that guarantees closeness to an elastica after the rectification of the block segmentation. We can use the same approach for Bézier curve 1, and we therefore have a design tool for elastic splines that includes a predetermined block segmentation.

5.3 The analytical description

If we want to cut a foliation of numerical elastic splines with a hot-blade robot, the robot needs the endpoints and end tangents of the rod that go beyond the curve segments of the numerical elastic splines. To get the endpoints and end tangents, we, therefore, extend the curve segments to the length of the rod. We extend the segments of a numerical elastic spline by determining the analytical description of the spline and extending the segments through the domain (this is possible because elastic curves have constant speed). To obtain the analytical description of the spline, we first determine the initial guesses of the curve segments with the algorithm described in Section 2.1.1. Because the curve segments are close to elastic curves, the initial guesses are close to the curve segments. We obtain an analytical C^0 elastic spline by scaling and rotating the initial guesses. To obtain the analytical description of the C^1 or C^2 elastic spline, we use the C^0 elastic spline as an initial guess and apply an optimization routine. That is, we minimize the L^2 distance between the numerical elastic spline and analytical elastic spline subject to the continuity constraints.

5.4 Design of surfaces foliated by elastic splines

Motivated by the hot-blade technology we extend the above mentioned methods for elastic splines to the design of surfaces that are foliated by elastic splines. Because a flexible rod takes the shape of an elastic curve, these designs are production-ready for the hot-blade technology. That is we can use a hot-blade robot to cut the design. In this section, we first extend the interpolation tool, described in Section 5.1, to the design of surfaces foliated by elastic splines. Next, in Section 5.4.2, we explain how we extend the data-driven design tool.

5.4.1 Design of surfaces with elastic splines obtained via interpolation and optimization

An easy way to generate surfaces foliated by elastic splines is to translate a single elastic spline along a curve. However, this approach only creates an unnecessarily restricted class of surfaces. To obtain more design freedom, we perform an interpolation of elastic splines such that a set of input elastic splines are connected continuously by other elastic splines. We do this by extending the tool from Section 5.1. To obtain a design of surface that can be cut with m blocks of foam, we propose the following three-step algorithm:

1. First, we draw n planar C^j, $j = 1$ or $j = 2$, cubic splines on the front, back, and inside m blocks, placed next to each other. We draw the cubic splines such that each spline contains m segments, each of which is contained in a separate block, see Figure 5.14.

2. By interpolating the n cubic splines with a cubic interpolation, we obtain a spline surface. Now, if we want to change the surface, we can go back to step 1 and modify the n cubic splines. This spline surface is our rough estimate of the production-ready m block design that we want to achieve.

3. The surface from step 2 has isocurves (isoparametric curves) in one direction given by C^j cubic splines. To create a surface foliated by numerical elastic splines, we apply the method from Section 5.1 to a large set of the isocurves. For $j = 1$ we get numerical C^1 elastic splines that interpolate the points and tangent angles at the block boundaries. For $j = 2$ we get numerical C^2 elastic splines that interpolate the points at the block boundaries but only the two end tangent angles. After we have applied the method, we obtain a production-ready m block design, described as m foliations of numerical elastic curves.

Once we have obtained the numerical elastic splines in step 3, we can get the analytical descriptions by means of the procedure described in Section 5.3.

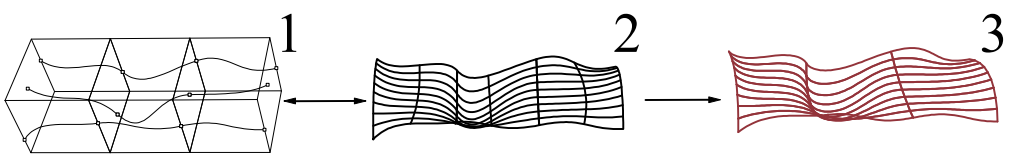


Fig. 5.14.: Step 1: the designer draws n (here 3) cubic splines. Step 2: the curves are interpolated and the designer can change the design. Step 3: the production-ready surface is visualized.

We have implemented the above algorithm as a tool in the CAD system Rhino 5. Using this tool, we can obtain foliations of elastic splines. Figures 5.15 and 5.16 display two examples, respectively a C^1 and C^2 design. In Figure 5.17 we consider the two examples and compare the cubic spline surfaces, from step 2, with the surfaces foliated by numerical elastic splines. Even though this interpolation tool can give us designs foliated by elastic splines, there are still disadvantages. We are not guaranteed that the cubic splines, step 2, are visually close to being elastic splines. Hence we might get numerical elastic splines that appear to foliate a non-smooth design or the output might deviate a lot from the design intent. In the next section we present a method that circumvents these problems. The method utilizes the data-driven design tool for elastic splines.

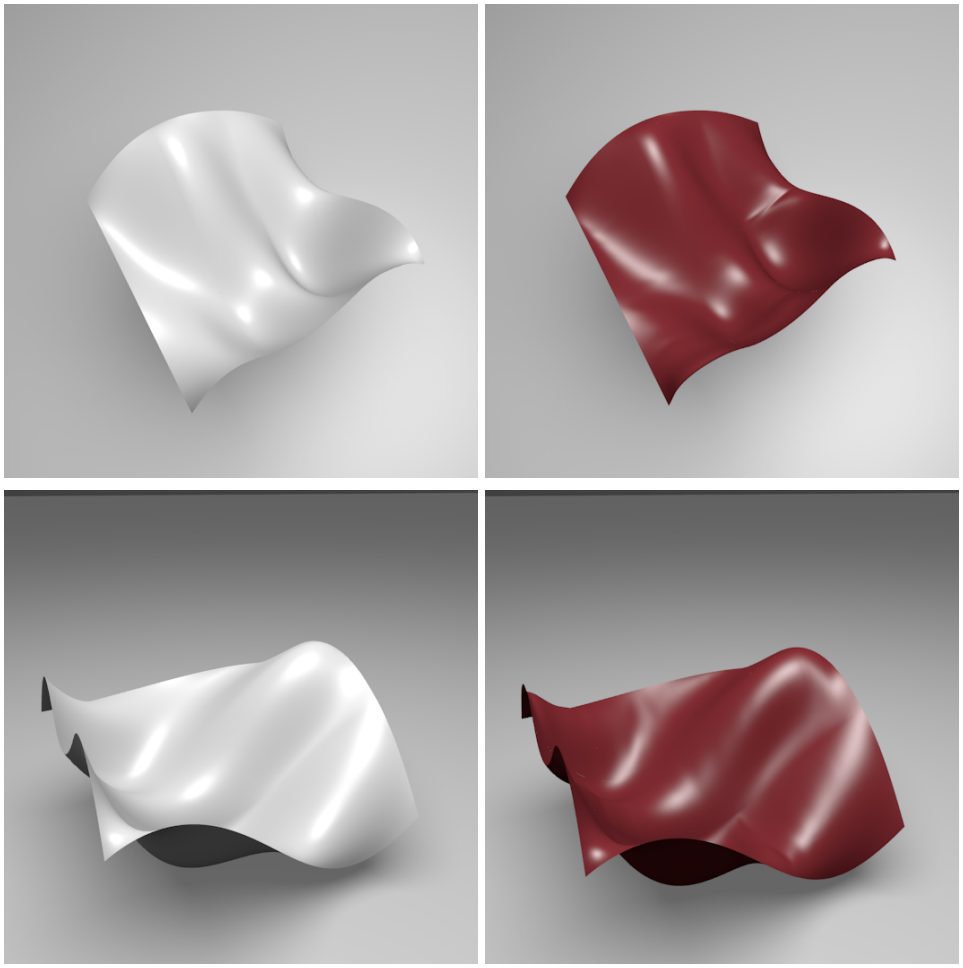


Fig. 5.15.: A 3×3 block C^1 design. Left (white): surface foliated by cubic splines. Right (red): surface foliated by numerical elastic splines.

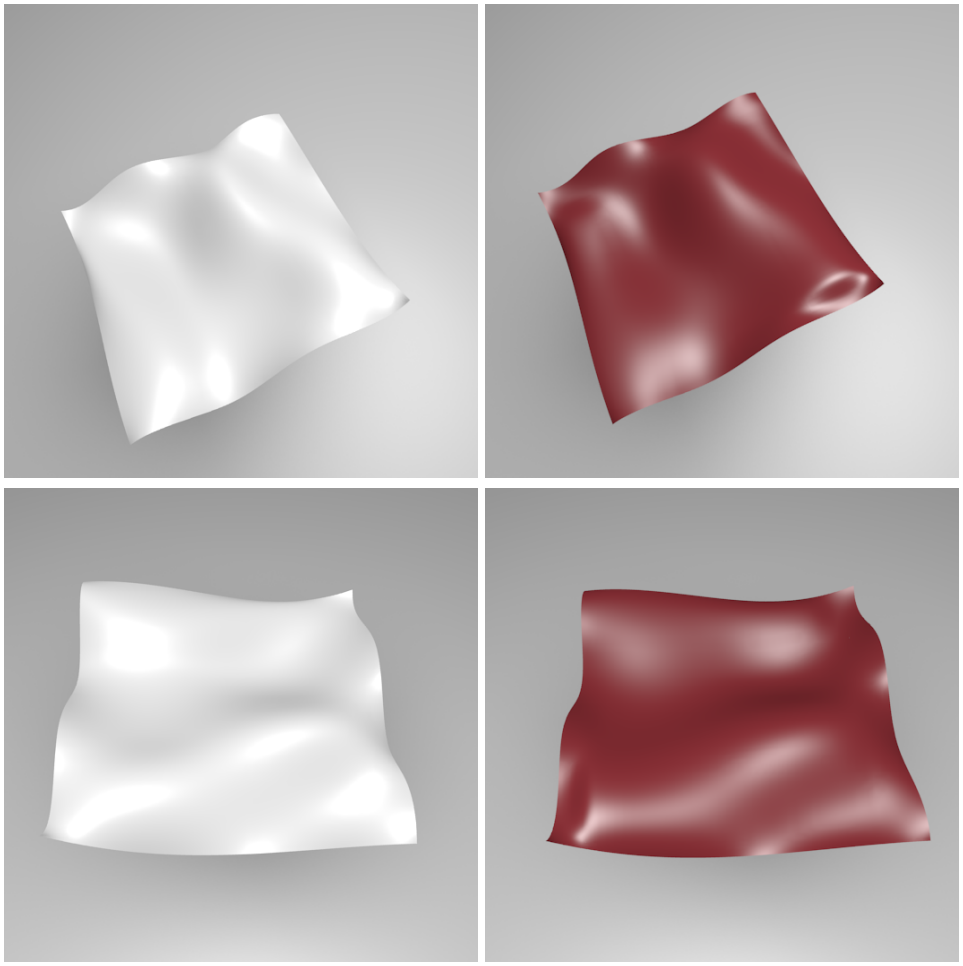


Fig. 5.16.: A 3×3 block C^2 design. Left (white): surface foliated by cubic splines. Right (red): surface foliated by numerical elastic splines.

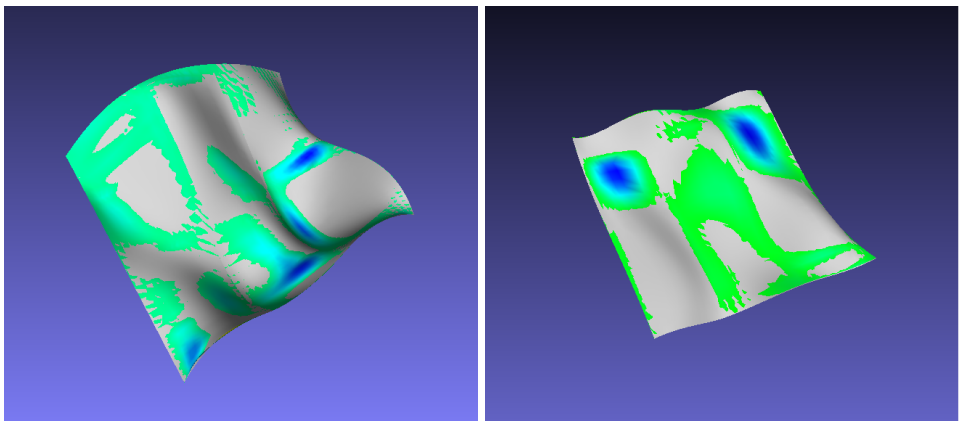


Fig. 5.17.: Comparison between the cubic spline surfaces and the designs foliated by (numerical) elastic splines (stacked on top of each other) in MeshLab [Cig+08]. The designs are made as 3×3 meter block designs. Left: C^1 design. The max closest point distance is 7.9 cm. Right: C^2 design, deviation is 9.3 cm.

5.4.2 Design of surfaces with elastic splines obtained via a database

To obtain an interactive design tool for surfaces foliated by elastic splines, we propose a method that extends the data-driven tool developed in Section 5.2. We use the same algorithmic framework as above: given m blocks we interpolate n planar cubic splines with m segments. But we restrict the control points of the n splines to the good regions determined by the database in Section 5.2 (and hence the cubic splines are visually close to elastic splines). This way we utilize the data-driven tool and avoid the limitations of the method above. We obtain interactivity, shape control, and we are guaranteed to get smooth foliations. To get the surface we use a cubic interpolation of the n input splines with the not-a-knot condition [Far02]. This gives us a cubic spline surface representation of the design. In most cases, the spline surface is production-ready, meaning that the splines between the n interpolated splines are also visually close to elastic splines. We can immediately determine this by computing the λ-residual for a dense set of curves in the interpolation. If one of the curves has a large λ-residual, we let the designer rectify the problem. The designer can rectify the problem in two ways. Either the designer changes the n input splines or he/she modifies the violating spline as another input spline. Once we have the production-ready design, represented as a spline surface, we obtain the analytical description of the spline foliation by means of the procedure in Section 5.3. In Figure 5.18 and 5.19 we include two examples of elastic spline foliations made with the interactive design tool.

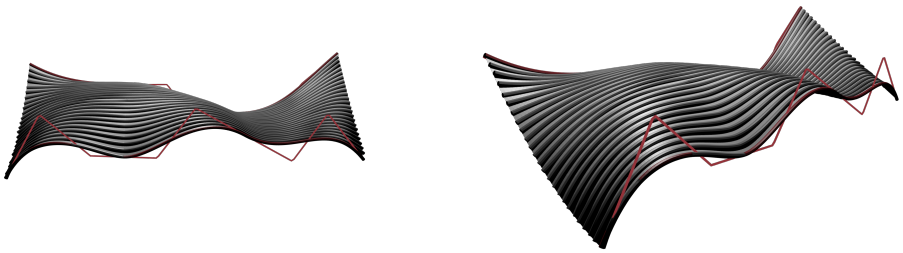


Fig. 5.18.: The production-ready design. The red control polygons are the control polygons of the cubic splines on the boundaries. The gray curves are the elastic splines.

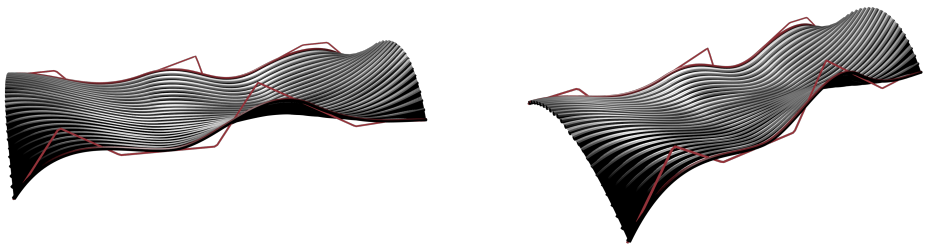
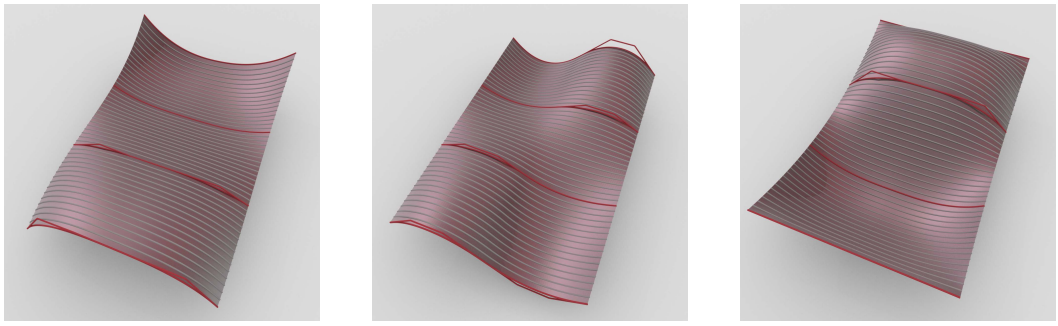


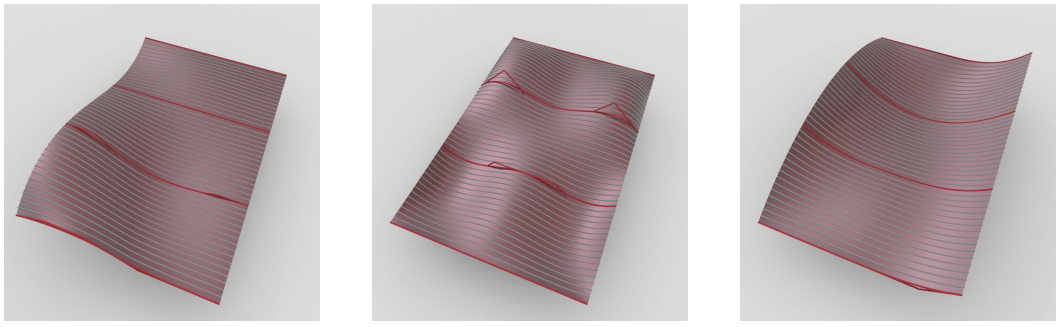
Fig. 5.19.: Another production-ready design.

To evaluate the data-driven design tool, we have created twelve different examples with the tool, displayed in Figures 5.20 to 5.23. We have created the examples with different design constraints to test if we can get a large variety of designs. The examples are generated with the tool by taking four input design curves (red) each of which has two cubic curve segments. The gray curves in Figures 5.20 to 5.23 are the elastic splines. For each of these designs, we have extracted 49 cubic splines in the interpolation and computed their λ-residuals. In Table 5.1 we summarize the results and obtain that the λ-residuals are below our tolerance (we used the tolerance 0.21 in the construction of the database).



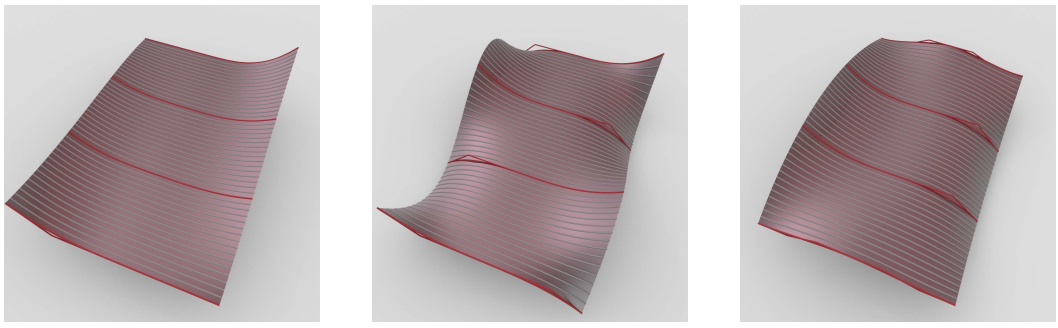
(1) (2) (3)

Fig. 5.20.: Foliations: from concave to convex.



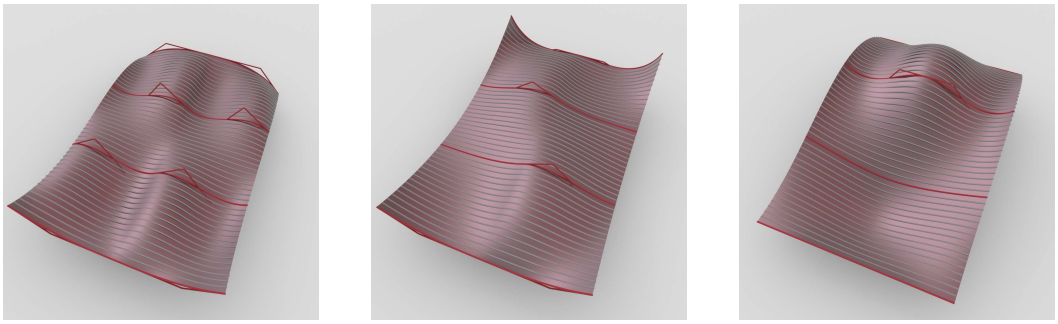
(4) (5) (6)

Fig. 5.21.: Foliations: fixed end(s).



(7) (8) (9)

Fig. 5.22.: Foliations: one end is moving up, the other end is moving down.



(10) (11) (12)

Fig. 5.23.: Foliations: same two boundary curves.

Tab. 5.1.: The maximal value of the λ-residuals for the four design curves and all of the 49 curves.

Foliation	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)
$\max(\lambda$-residual) (Design curves)	0.06	0.10	0.04	0.03	0.11	0.05	0.06	0.03	0.09	0.12	0.10	0.16
$\max(\lambda$-residual) (All curves)	0.08	0.10	0.07	0.06	0.14	0.06	0.06	0.04	0.09	0.17	0.10	0.18

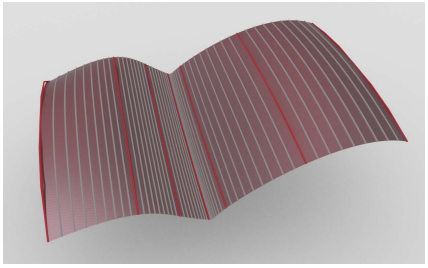
5.4.3 Hot-blade cutting context

Having the surface tools, we are able to obtain production-ready designs for the hot-blade technology. We have tested the data-driven tool in a hot-blade cutting context by considering a 2×3 block design, made by architect Zeynep Bacinoglu at an internal summer school, see Figure 5.24a. Given this design, we have reconstructed it with the tool using the red input cubic splines in Figure 5.24b. These splines contain two curve segments, where each segment is inside of a block. After the cubic interpolation of the splines, we extracted 55 splines from the spline surface and computed the L^2 distance and Hausdorff distance to the approximating elastic splines. We summarize the results in Table 5.2 where the Hausdorff distance is measured in mm and the L^2 distance is normalized with the curve length. From these results, we observe that the splines get very close to the elastic spline approximation. In Figure 5.24c we compare the design curves with structured light scans of the fabricated design. The picture of the scan shows that the design curves are close to the final product.

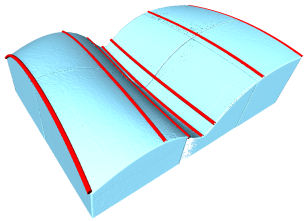
Tab. 5.2.: The λ-residual and distances for the design curves and all curves, Figure 5.24.

Curves	max Design	max All	mean Design	mean All	median Design	median All	variance Design	variance All
λ-residual	7.86e-03	8.72e-03	1.61e-03	1.20e-03	3.28e-04	2.80e-04	7.32e-06	4.89e-06
L^2 distance	4.52e-05	4.52e-05	9.60e-06	6.53e-06	6.36e-07	3.56e-07	2.99e-10	1.36e-10
Hausdorff distance	7.84e-02	7.84e-02	2.69e-02	2.25e-02	1.39e-02	1.39e-02	6.14e-04	2.96e-04

(a) The fabricated six block design. The block dimension is $550 \times 550 \times 500$ mm.



(b) The design curves (red) and elastic spline foliation (gray).



(c) Structured light scans of the six blocks with the design curves.

Fig. 5.24.: Design of architectural formwork.

In this chapter we have used the word *production-ready* for designs that are foliated by elastic splines. However, to be truly production-ready, we must address all the production constraints listed in Section 1.2.

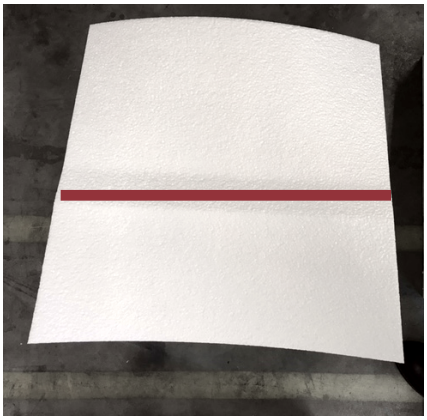
Fig. 5.25.: An example of a design with a ridge caused by an unstable rod configuration.

In Section 1.2 most of the production constraints can be resolved by a workaround strategy (i.e., change of hardware, robotic setup, etc.) but the unstable rod configurations are not easily dealt with. An unstable configuration makes the rod wobble and consequently the final design will have unwanted ridges/bumps, see Figure 5.25. In most of our test cases, the unstable curves happened when the rod took the shape of a line. Instead of avoiding designs with straight lines, we propose solutions for rationalizing an elastic curve foliation that contains a line. We assume the foliation is created with one of our design tools, and hence the curves in the foliation are on parallel planes to the block boundary, see Figure 5.26b. To first detect the line we consider the curvature of the curves. Once we have detected a line, we propose two solutions for obtaining a line-free foliation of the design:

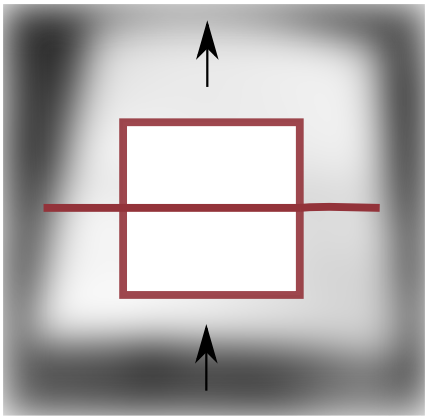
Solution 1. We rotate the planes, parallel to the block boundary, by the same amount. Figure 5.26c illustrates this. The intersections of the rotated planes with the design give us a new set of curves that foliate the design. To get the elastic curve foliation, we apply an elastic curve approximation.

Solution 2. Instead of rotating the planes by the same amount (Solution 1), we apply two rotations of the planes: counterclockwise and clockwise. The centers of rotation are left and right to respectively the front and back side of the block, see Figure 5.26d.

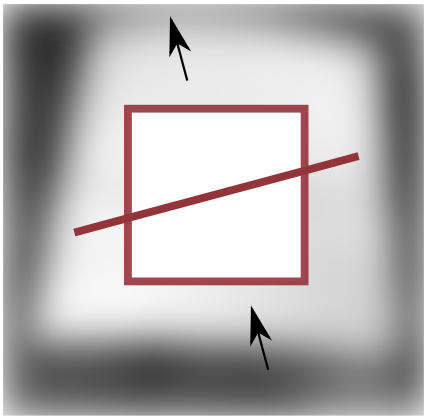
In both cases, we determine the amount of rotation needed using a tolerance for not being close to a line. Given this tolerance, we increase the amount of rotation in Solutions 1–2 until the new foliation does not contain a line.



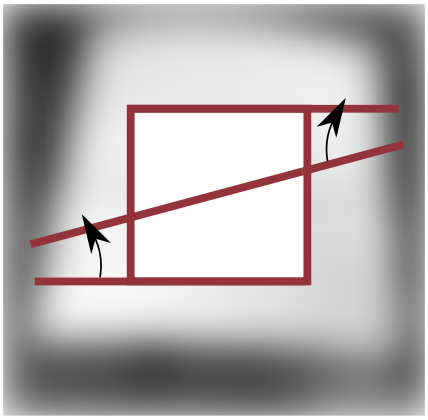
(a) The line causes a ridge in the final design (at the red line).



(b) Original foliation. Planes parallel to the block boundary.



(c) Solution 1: we rotate the planes with the same amount.



(d) Solution 2: we apply two rotations.

Fig. 5.26.: Rationalization of a design that contains a line (top view).

6 Conclusion

In this Ph.D. project, we have considered the problem of developing design tools for creating elastic curves, elastic splines, and surfaces foliated by elastic splines, in particular, production-ready designs for robotic hot-blade cutting. Existing methods in the literature provide us with tools for obtaining interpolating elastic splines (specified by a sequence of points with/without tangents). These methods can, in principle, be used as design tools for elastic splines, but have several issues. Due to the non-uniqueness of a solution and the need for an optimization routine/iteration scheme, the methods lack shape control, can be unreliable, and are non-interactive. Hence the methods fall short from a practical standpoint. In this project, we therefore take a different approach. Instead of determining an elastic spline given by a sequence of points and tangents, we impose restrictions on cubic curves and splines and use them as proxies for elastic curves and splines. This approach gives us not only design and projection tools for elastic curves and splines but also extends to design tools for surfaces foliated by elastic splines. With these tools, we obtain a supplement to the existing NURBS library that provides us with designs of elastic curves, elastic splines, and surfaces swept by elastic splines/production-ready designs for hot-blade cutting.

We hope that the contribution of the project can facilitate the employment of new fabrication technologies, especially hot-blade cutting, and make people able to create fast and inexpensive curved building elements that otherwise would be too expensive to fabricate. From a methodology perspective, we hope to inspire people to utilize the advantages of the polynomial splines and include production constraints directly into CAD systems via restrictions on the control polygons.

6.1 Test cases

The conducted Ph.D. is part of the larger project: Digital Factory. Because Digital Factory is a collaborative project between the Technical University of Denmark, a robotics company, and an architectural practice, we have been able to test the algorithms in practice at various workshops and an internal summer school. At the summer school, the architects at GXN made a large wavy wall using a Grasshopper script that incorporated the projection tool, see Figure 6.3. But due to the unstable configurations of the rod, the wall exhibited unfortunate bumps/ridges. However,

a more recent 12 block design was made without unstable rod configurations, see Figure 6.1.

Fig. 6.1.: Left: the 12 block design. Center: the design and the inverse cut. Right: evaluating continuity.

6.2 Future work

We propose the following as future work:

- to determine the number of connected components of the cubic Bézier curves that have small λ-residuals, see Section 3.4,
- to classify all unstable configurations of the rod, and
- to consider the problem of designing 3-D elastic curves/splines with a spline interface.

A different, but less mathematical, suggestion for future work is to attach sensors to a rod that send the boundary values to a script that almost instantaneously solves the boundary value problem, see Figure 6.2.

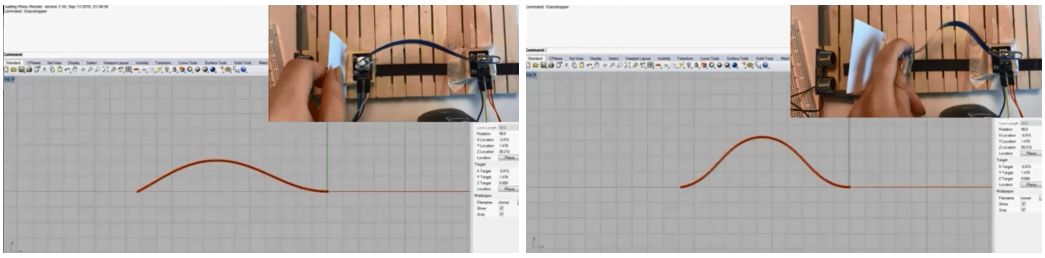


Fig. 6.2.: A prototype for a direct CAD representation of a flexible rod using an ultra sonic sensor and a rotary encoder.

Fig. 6.3.: A large wall made with robotic hot-blade cutting.

Bibliography

[Ard18] Andrei A. Ardentov. „Multiple Solutions in Euler's Elastic Problem". In: *Automation and Remote Control* 79 (2018), pp. 1191–1206 (cit. on p. 70).

[BG17] David Brander and Jens Gravesen. „Surfaces foliated by planar geodesics: a model for curved wood design". In: *Proceedings of Bridges 2017*. Tessellations Publishing, 2017, pp. 487–490 (cit. on pp. 7, 11).

[BK94] Guido Brunnett and Johannes Kiefer. „Interpolation with Minimal-energy Splines". In: *Computer-Aided Design* 26.2 (1994), pp. 137–144 (cit. on pp. 65, 66).

[Bor06] Max Born. „Untersuchungen über die Stabilität der elastische Linie in Ebene und Raum". PhD thesis. 1906 (cit. on p. 11).

[Bra+16] David Brander, J. Andreas Bærentzen, Kenn Clausen, et al. „Designing for hot-blade cutting: Geometric Approaches for High-Speed Manufacturing of Doubly-Curved Architectural Surfaces". In: *Advances in Architectural Geometry 2016*. vdf Hochschulverlag AG an der ETH Zürich, 2016, pp. 306–327 (cit. on pp. xi, 4).

[Bra+17] David Brander, Jens Gravesen, and Toke B. Nørbjerg. „Approximation by planar elastic curves". In: *Advances in Computational Mathematics* 43 (2017), pp. 25–43 (cit. on pp. 7, 13, 15, 21, 27, 38).

[Bra+18a] David Brander, J. Andreas Bærentzen, Ann-Sofie Fisker, and Jens Gravesen. „Bézier curves that are close to elastica". In: *Computer-Aided Design* 104 (2018), pp. 36–44 (cit. on pp. xi, 3, 4).

[Bra+18b] David Brander, Jakob Andreas Bærentzen, Ann-Sofie Fisker, and Jens Gravesen. „Designing interactively with elastic splines". In: *Computer Aided Geometric Design* 62 (2018), pp. 181–191 (cit. on pp. xi, 3, 4).

[Bru+01] Alfred M. Bruckstein, Arun N. Netravali, and Tom J. Richardson. „Epi-convergence of discrete elastica". In: *Applicable Analysis* 79.1-2 (2001), pp. 137–171 (cit. on p. 15).

[Bru+96] Alfred M. Bruckstein, Robert J. Holt, and Arun N. Netravali. „Discrete elastica". In: *Discrete Geometry for Computer Imagery*. Springer Berlin Heidelberg, 1996, pp. 59–72 (cit. on p. 15).

[Cig+08] Paolo Cignoni, Marco Callieri, Massimiliano Corsini, et al. „MeshLab: an Open-Source Mesh Processing Tool". In: *Eurographics Italian Chapter Conference*. The Eurographics Association, 2008, pp. 129–136 (cit. on p. 79).

[Clo17] CloudCompare. *CloudCompare*. www.cloudcompare.org, 2017 (cit. on p. 9).

[CP99] Horng-Yang Chen and Helmut Pottmann. „Approximation by ruled surfaces“. In: *Journal of Computational and Applied Mathematics* 102 (1999), pp. 143 –156 (cit. on p. 7).

[Ede+83] Herbert Edelsbrunner, David Kirkpatrick, and Raimund Seidel. „On the shape of a set of points in the plane“. In: *IEEE Transactions on Information Theory* 29.4 (1983), pp. 551–559 (cit. on p. 73).

[Edw92] John A. Edwards. „Exact Equations of the Nonlinear Spline“. In: *ACM Transactions on Mathematical Software* 18.2 (1992), pp. 174–192 (cit. on p. 65).

[Eul44] Leonhard Euler. „Additamentum I: De curvis elasticis“. In: *Methodus inveniendi lineas curvas maximi minimive proprietate gaudentes*. 1744 (cit. on p. 11).

[Far02] Gerald Farin. „A history of curves and surfaces in CAGD“. In: *Handbook of Computer Aided Geometric Design*. North-Holland, 2002, pp. 1–21 (cit. on pp. 12, 25, 80).

[Flö+12] Simon Flöry, Yukie Nagai, Florin Isvoranu, Helmut Pottmann, and Johannes Wallner. „Ruled Free Forms“. In: *Advances in Architectural Geometry 2012*. Ed. by Lars Hesselgren, Shrikant Sharma, Johannes Wallner, et al. Springer Vienna, 2012, pp. 57–66 (cit. on p. 7).

[FP10] Simon Flöry and Helmut Pottmann. „Ruled Surfaces for Rationalization and Design in Architecture“. In: *LIFE in:formation. On Responsive Information and Variations in Architecture*. Association for Computer Aided Design in Architecture (ACADIA), 2010, pp. 103–109 (cit. on p. 7).

[Gla66] Jeremy M. Glass. „Smooth-curve interpolation: A generalized spline-fit procedure“. In: *BIT Numerical Mathematics* 6.4 (1966), pp. 277–293 (cit. on p. 65).

[Hac+14] Norman Hack, Willi V. Lauer, Fabio Gramazio, and Matthias Kohler. „Mesh-Mould: Differentiation for Enhanced Performance“. In: *Rethinking Comprehensive Design*. CAADRIA, 2014, pp. 139–148 (cit. on p. 8).

[Hor83] Berthold K. P. Horn. „The Curve of Least Energy“. In: *ACM Transactions on Mathematical Software* 9.4 (1983), pp. 441–460 (cit. on p. 15).

[HS98] Josef Hoschek and Ulrich Schwanecke. „Interpolation and approximation with ruled surfaces“. In: *Mathematics of surfaces, VIII*. Information Geometers, 1998, pp. 213–232 (cit. on p. 7).

[Law13] Derek F. Lawden. *Elliptic Functions and Applications*. Springer New York, 2013 (cit. on p. 96).

[Lev08] Raph Levien. *The elastica: a mathematical history*. Tech. rep. UCB/EECS-2008-103. EECS Department, University of California, Berkeley, 2008 (cit. on p. 11).

[Lov34] Augustus E.H. Love. *A Treatise on the Mathematical Theory of Elasticity*. University Press, 1934 (cit. on p. 11).

[Mal77] Michael A. Malcolm. „On the Computation of Nonlinear Spline Functions“. In: *SIAM Journal on Numerical Analysis* 14.2 (1977), pp. 254–282 (cit. on p. 65).

[MAT16] MATLAB. *(R2016b)*. The MathWorks, Inc., 2016 (cit. on p. 27).

[Meh74] Even Mehlum. „Nonlinear Splines“. In: *Computer Aided Geometric Design (Proc. Conf., Univ. Utah, Salt Lake City, Utah, 1974)*. Academic Press, 1974, pp. 173–207 (cit. on pp. 65, 66).

[Mun00] James R. Munkres. *Topology*. Prentice Hall, 2000 (cit. on p. 45).

[NW06] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer New York, 2006 (cit. on pp. 12, 17, 27).

[Nør16] Toke B. Nørbjerg. „Rationalization in architecture with surfaces foliated by elastic curves“. PhD thesis. 2016 (cit. on pp. 12, 95).

[PT97] Les Piegl and Wayne Tiller. *The NURBS Book*. Springer New York, 1997 (cit. on p. 19).

[Rau+12] Christian Raun, Mathias K. Kristensen, and Poul H. Kirkegaard. „Dynamic Double Curvature Mould System“. In: *Computational Design Modelling: Proceedings of the Design Modelling Symposium Berlin 2011*. Springer Berlin Heidelberg, 2012, pp. 291–300 (cit. on p. 8).

[Rie03] Eric Rieth. „First Archaeological Evidence of the Mediterranean Whole Moulding Ship Design Method: The Example of the Culip VI Wreck, Spain (XIIIth-XIVth c.)“ In: *Shipbuilding Practice and Ship Design Methods From the Renaissance to the 18th Century*. Max Planck Institute for the History of Science, 2003, pp. 9–16 (cit. on p. 65).

[Rus+16] Romana Rust, Fabio Gramazio, and Matthias Kohler. „Force Adaptive Hot-Wire Cutting“. In: *Advances in Architectural Geometry 2016*. vdf Hochschulverlag AG an der ETH Zürich, 2016, pp. 288–305 (cit. on p. 8).

[SO18] Yusuke Sakai and Makoto Ohsaki. „Discrete elastica for shape design of grid-shells“. In: *Engineering Structures* 169 (2018), pp. 55–67 (cit. on p. 11).

[Ste+16] Kasper H. Steenstrup, Toke B. Nørbjerg, Asbjørn Søndergaard, J. Andreas Bærentzen, and Jens Gravesen. „Cuttable Ruled Surface Strips for Milling“. In: *Advances in Architectural Geometry 2016*. vdf Hochschulverlag AG an der ETH Zürich, 2016, pp. 328–342 (cit. on p. 7).

[Woo69] Chris H. Woodford. „Smooth curve interpolation“. In: *BIT Numerical Mathematics* 9.1 (1969), pp. 69–77 (cit. on p. 65).

[Wu+16] Peng Wu, Jun Wang, and Xiangyu Wang. „A critical review of the use of 3-D printing in the construction industry“. In: *Automation in Construction* 68 (2016), pp. 21–31 (cit. on p. 8).

[Yam13] Fujio Yamaguchi. *Curves and Surfaces in Computer Aided Geometric Design*. Springer Berlin Heidelberg, 2013 (cit. on p. 26).

A

Appendix: elliptic functions and the extension of the k parameter

There are twelve Jacobian elliptic functions and the basic functions cn, sn, and dn are the most commonly used functions (respectively elliptic cosine, elliptic sine, and the delta amplitude). There are several ways of defining cn, sn, and dn. To define the functions we introduce the incomplete elliptic integral of the first kind for $k \in [0, 1]$:

$$F(\phi, k) = \int_0^\phi \frac{\mathrm{d}\theta}{\sqrt{1 - k^2 \sin^2 \theta}} .$$

If we let $u = F(\phi, k)$ then $\phi = F^{-1}(u, k) = \mathrm{am}(u, k)$ where am is referred to as the elliptic amplitude. With am we obtain the basic elliptic functions as

$$\begin{aligned} \mathrm{cn}(u, k) &= \cos \mathrm{am}(u, k) , \\ \mathrm{sn}(u, k) &= \sin \mathrm{am}(u, k) , \\ \mathrm{dn}(u, k) &= \sqrt{1 - k^2 \sin^2 \mathrm{am}(u, k)} . \end{aligned}$$

We use dn to obtain the second incomplete integral $E(\phi, k)$:

$$E(\phi, k) = \int_0^\phi \mathrm{dn}^2(u, k)\, \mathrm{d}u.$$

If we evaluate E and F at $\pi/2$ we get respectively the second and first complete elliptic integral. The first complete elliptic integral $K(k) = F(\pi/2)$ is also called a quarter period because the elliptic functions satisfy

$$\begin{aligned} \mathrm{cn}(u + 2K, k) &= -\mathrm{cn}(u, k) , \\ \mathrm{sn}(u + 2K, k) &= -\mathrm{sn}(u, k) , \\ \mathrm{dn}(u + 2K, k) &= \mathrm{dn}(u, k) . \end{aligned}$$

Employing the elliptic functions and integrals, one can obtain the parameterization of an elastic curve [Nør16]. Up to scaling, translation, and rotation of the ambient space, the inflectional elastic curves are given by

$$\xi_k(s) = \begin{pmatrix} 2E(s, k) - s \\ 2k(1 - \mathrm{cn}(s, k)) \end{pmatrix} \tag{A.1}$$

for $k \in [0,1)$. For the non-inflectional case we obtain

$$\xi_k(s) = \begin{pmatrix} (1-\frac{2}{k^2})s + \frac{2}{k}E(\frac{s}{k},k) \\ \frac{2}{k}\left(1-\mathrm{dn}(\frac{s}{k},k)\right) \end{pmatrix} \tag{A.2}$$

for $k \in [0,1]$. In Chapter 2 we use (A.1) to describe the non-inflectional elastica (A.2). This is possible if we extend k to $[0,\infty)$. For $k > 1$ we have the analytic continuations [Law13]:

$$\begin{aligned} \mathrm{cn}(u,k) &= \mathrm{dn}\left(ku, \frac{1}{k}\right), \\ \mathrm{sn}(u,k) &= \frac{1}{k}\mathrm{sn}\left(ku, \frac{1}{k}\right), \\ \mathrm{dn}(u,k) &= \mathrm{cn}\left(ku, \frac{1}{k}\right), \\ E(u,k) &= kE\left(ku, \frac{1}{k}\right) + u(1-k^2). \end{aligned}$$

With these extensions, we can write all elastica (up to scaling, translation, and rotation of the ambient space) as (A.1) where $k \geq 1$ gives us the non-inflectional elastic curves.

B

Appendix: design of elastic curves

We took ten elastic curve segments and extracted their boundary values (endpoints, end tangent angles, and length). Using the boundary values, we determined cubic Bézier curves with the same endpoints, end tangent angles, and length. To these curves, we applied Method 2, Chapter 2, and computed the bending energy after 0,1,...,6 knot updates. In Table B.1 we summarize the results. The first number in the convergence sequence is the bending energy before applying the optimization.

Tab. B.1.: Solving the boundary value problem with Method 2. The bending energy was computed after 0,1,...,6 knot updates.

Example	The elastic curve	The Bézier curve	Convergence sequence (bending energy)
1			(0.08832, 0.08809, 0.08778, 0.08769, 0.08769, 0.08769,0.08769,0.08769)
2			(0.4751, 0.47337 ,0.45771, 0.45049,0.44783, 0.44772, 0.44770, 0.44770)
3			(0.05407, 0.05259, 0.05252, 0.05252,0.05252, 0.05251, 0.05251, 0.05251)
4			(0.08848, 0.08848, 0.08802, 0.08799, 0.08798, 0.08798, 0.08798, 0.08798)
5			(0.1513, 0.15061, 0.15018,0.14975, 0.14973, 0.14973, 0.14973, 0.14973)
6			(0.1467, 0.14207, 0.14040, 0.14016, 0.14014, 0.14014, 0.14014, 0.14014)
7			(0.1520, 0.14900, 0.14746 0.14730 0.14730, 0.14730, 0.14730 0.14730)
8			(0.25120, 0.24687, 0.24519, 0.24426,0.24413, 0.24412, 0.24412, 0.24412)
9			(0.13310, 0.13309, 0.13237, 0.13170, 0.13160, 0.13160, 0.13158, 0.13158)
10			(1.005, 0.68234, 0.66807, 0.64544, 0.64505, 0.64500, 0.64500, 0.64500)

Index